TP WEB SÉMANTIQUE (Conception d'un graphe de connaissances)

PROJET : Modélisation des connaissances dans le domaine des Produits alimentaires avec un graphe de connaissances

Réalisé par:

Nom & prenom	email
Salimata cissouma	salimata.cissouma@etu.univ-grenoble-alpes.fr
MOHAMMEDI Amira	amira.mohammedi@etu.univ-grenoble-alpes.fr
Gaelle Manuela Yonga Yonga	gaelle.yonga-yonga@etu-univ

sous la supervision de :

Danielle

TP WEB SEMANTIQUE

1. **But**

La modélisation des connaissances des produits alimentaires permet de structurer et d'analyser de manière détaillée les informations liées aux produits alimentaires, à leurs ingrédients, à leur valeur nutritionnelle, à leurs propriétés, et à leur traçabilité. Un graphe de connaissances dans ce contexte peut être extrêmement utile pour des applications dans la gestion des produits, la recherche de recettes, la recommandation alimentaire, l'étiquetage, ainsi que pour des analyses de tendances et des études sur la santé publique.

2. **Objectifs de la modélisation des connaissances**

- **Structurer les informations alimentaires** : Créer une base de données sémantique permettant d'intégrer des données sur les produits alimentaires, les ingrédients, leurs propriétés nutritionnelles, leurs relations avec la santé et l'environnement.
- **Améliorer la traçabilité et la transparence** : Relier les produits aux informations relatives à leur origine (produits biologiques, traçabilité des chaînes d'approvisionnement, méthodes de production, etc.).
- **Faciliter la recherche et les recommandations** : Offrir des fonctionnalités de recherche avancées pour trouver des produits en fonction de critères spécifiques comme les préférences alimentaires, les restrictions nutritionnelles (sans gluten, végétalien, etc.), ou les valeurs environnementales (énergie, empreinte carbone, etc.).
- **Analyser les tendances et la consommation** : Utiliser les relations sémantiques pour comprendre les habitudes de consommation, les tendances alimentaires, et les impacts sur la santé publique.

3. Compréhension et préparation des Données

Pour notre projet, nous utilisons la source de données du site Open food facts (https://world.openfoodfacts.org/data **).** Open food facts étant un projet web sémantique met à disposition d'un jeu de données au format CSV, d'un fichier RDF très volumineux. Nous réalisons les investigations suivantes:

❖ Identification des concepts et des variables

Les informations sur les entites du fichier CSV sont fournis dans un fichier texte (https://static.openfoodfacts.org/data/data-fields.txt), permettant de prendre connaissance du type de valeurs de chaque entite et d'identifier les concepts et les variables necessaires pour la conception du graphe de connaissance.

Les **entités principales** considérées dans notre cas d'étude incluent :

- **Produit alimentaire** : le concept central, représentant les aliments eux-mêmes.
- **Catégorie** : les groupes auxquels les produits appartiennent (fruits, légumes, boissons, etc.).
- **Marque** : les marques associées aux produits.
- **Ingrédients** : les composants du produit (e.g., sucre, sel, farine).
- **Valeur Nutritionnelle** : informations sur les calories, protéines, lipides, etc.
- **Allergène** : substances allergènes (gluten, arachides, etc.).
- **Origine** : la provenance du produit (pays, région).
- **Magasin** ou **Point de Vente** : lieux où les produits sont disponibles.

Les variables étant les **attributs descriptifs** ou les **propriétés** des concepts. Pour chaque concept, nous avons ressorti dans notre source de données les variables suivantes:

- Produit Alimentaire: code, product_name, creator, url(url of the product page on Open Food Facts), brands, allergens, ingredients_text, quantity, image_ingredients_url, created_datetime.
- Valeur Nutritionnelle : energy_100g, fat_100g, saturated-fat_100g, carbohydrates_100g, sugars_100g, fiber_100g, proteins_100g, salt_100g, carbon-footprint_100g.
- Origine : origins, countries, cities
- Magasin/ Point de Vente : manufacturing_places, purchase_places

❖ Identification des relations entre les concepts

Dans un graphe de connaissances, les concepts sont liés par des relations sémantiques. **les relations possibles dans notre contexte :**

- **"Produit" a pour marque** `'hasBrand'` **→ "marque"**
- **"Produit" a pour categorie** ‘`hasCategory`’ **→ "Categorie"**
- **"Produit" contient → "Ingrédient"**
- **"Produit Alimentaire" a pour valeur nutritionnelle** ‘`hasNutrient`’**→ "Valeur Nutritionnelle"**
- **"Produit" est valable dans “`availableInCountry`” → "pays"**
- **"Produit" a pour createur “**`hasCreator`**”→ "Createur"**
- **"Produit" contient** ‘`containsAllergen`’**→ "Allergène"**
- **"Produit" est disponible dans → "`PurchasePlace`"**
- **"Produit Alimentaire" est fabriqué en → "`ManufacturingPlace`"**
- **etc …**

❖ Préparation des Données

Pour pouvoir exploiter la source de données à notre disposition, nous avons fait les actions ci-dessous:

- Partitionnement: Le fichier csv contenant les différentes valeurs étant volumineuse, nous avons scindé en partition de plusieurs fichiers csv de 10000 lignes maximum par fichier, y compris les en-têtes. Facilitant ainsi son utilisation/sa manipulation sur une machine de petite capacité.
- Extraction des colonnes qui correspondent à nos variables d'intérêt.
- Nettoyage : Supprimer les doublons, corriger les valeurs manquantes ou erronées

4. Conception de l'ontologie

Le site de Open food facts fournit déjà un graphe RDF, mais le fichier contenant ce graphe étant volumineux, nous avons opté pour la création de notre propre Ontologie en employant deux approches:

- le logiciel protégé et
- les bibliothèques python adapté à la mise sur pied d'une ontologie.

Les vocabulaires comme FOAF (friend of a friend) et Schema.org ont été utilisés pour la démarche, afin de garantir l'interopérabilité.
Un Aperçu du processus de conception de notre ontologie à l'aide des bibliothèques python:

- Initialisation

```
import csv
import urllib.parse
import requests
from rdflib import Graph, Namespace, URIRef, Literal, RDF, RDFS
from rdflib.namespace import OWL, XSD, FOAF, DC
from SPARQLWrapper import SPARQLWrapper, JSON

# Chemins des fichiers
txt_file_path = "important_columns.txt"  # Fichier texte contenant les colonnes importantes
rdf_file_path = "output_ontology2.rdf"  # Fichier de sortie OWL

# Espaces de noms pour l'ontologie
SCHEMA = Namespace("http://schema.org/")
DBO = Namespace("http://dbpedia.org/ontology/")
EX = Namespace("http://produitsalimentaires.fr/")
FOAF = Namespace("http://xmlns.com/foaf/0.1/")
XSD = Namespace("http://www.w3.org/2001/XMLSchema#")

# Charger la liste des colonnes importantes depuis le fichier texte
try:
    with open(txt_file_path, "r") as file:
        important_columns = [line.strip() for line in file.readlines() if line.strip()]
except Exception as e:
    print(f"Erreur lors du chargement des colonnes : {e}")
    exit()

# Liaison des namespaces au graphe
g = Graph()
g.bind("schema", SCHEMA)
g.bind("dbo", DBO)
g.bind("foaf", FOAF)
g.bind("dc", DC)
g.bind("ex", EX)
g.bind("owl", OWL)
```

- Définition de l'ontologie et des classes

```
# Définition de l'ontologie
g.add((URIRef("http://produitsalimentaires.fr/foaf-ontology"), RDF.type, OWL.Ontology))
g.add((URIRef("http://produitsalimentaires.fr/foaf-ontology"), RDFS.comment, Literal("Une ontologie")))

# Définir les classes principales et sous-classes
g.add((EX.Product, RDF.type, OWL.Class))
g.add((DBO.Product, RDF.type, OWL.Class))

g.add((EX.Brand, RDF.type, OWL.Class))
g.add((DBO.Brand, RDF.type, OWL.Class))

g.add((EX.Category, RDF.type, OWL.Class))
g.add((DBO.Category, RDF.type, OWL.Class))

g.add((EX.Nutrient, RDF.type, OWL.Class))
g.add((EX.Allergen, RDF.type, OWL.Class))

g.add((EX.Creator, RDF.type, OWL.Class))
g.add((EX.ManufacturingPlace, RDF.type, OWL.Class))
g.add((EX.Ingredient, RDF.type, OWL.Class))
g.add((EX.PurchasePlace, RDF.type, OWL.Class))
g.add((EX.City, RDF.type, OWL.Class))
```

- Definition des proprietes

```
# Définir les propriétés et leur domaine et plage
g.add((EX.hasBrand, RDF.type, OWL.ObjectProperty))
g.add((EX.hasBrand, RDFS.domain, EX.Product))
g.add((EX.hasBrand, RDFS.range, EX.Brand))
g.add((EX.Country, RDF.type, OWL.Class))
g.add((DBO.Country, RDF.type, OWL.Class))

g.add((EX.hasCategory, RDF.type, OWL.ObjectProperty))
g.add((EX.hasCategory, RDFS.domain, EX.Product))
g.add((EX.hasCategory, RDFS.range, EX.Category))


g.add((EX.containsAllergen, RDF.type, OWL.ObjectProperty))
g.add((EX.containsAllergen, RDFS.domain, EX.Product))
g.add((EX.containsAllergen, RDFS.range, EX.Allergen))

g.add((EX.hasNutrient, RDF.type, OWL.ObjectProperty))
g.add((EX.hasNutrient, RDFS.domain, EX.Product))
g.add((EX.hasNutrient, RDFS.range, EX.Nutrient))


g.add((EX.availableInCountry, RDF.type, OWL.ObjectProperty))
g.add((EX.availableInCountry, RDFS.domain, EX.Product))
g.add((EX.availableInCountry, RDFS.range, DBO.Country))
```

Aperçu de la création de notre ontologie, en utilisant le logiciel protégé

Fournir une capture

5. Peuplement de l'ontologie

Pour peupler notre ontologie, on a utilisé notre jeu de données nettoyées contenus dans un fichier csv, afin de créer le graphe RDF tout en s'appuyant sur le vocabulaire défini. Cette étape transforme les données structurées (comme un fichier CSV) en triplets RDF interconnectés et conformes à la sémantique définie.

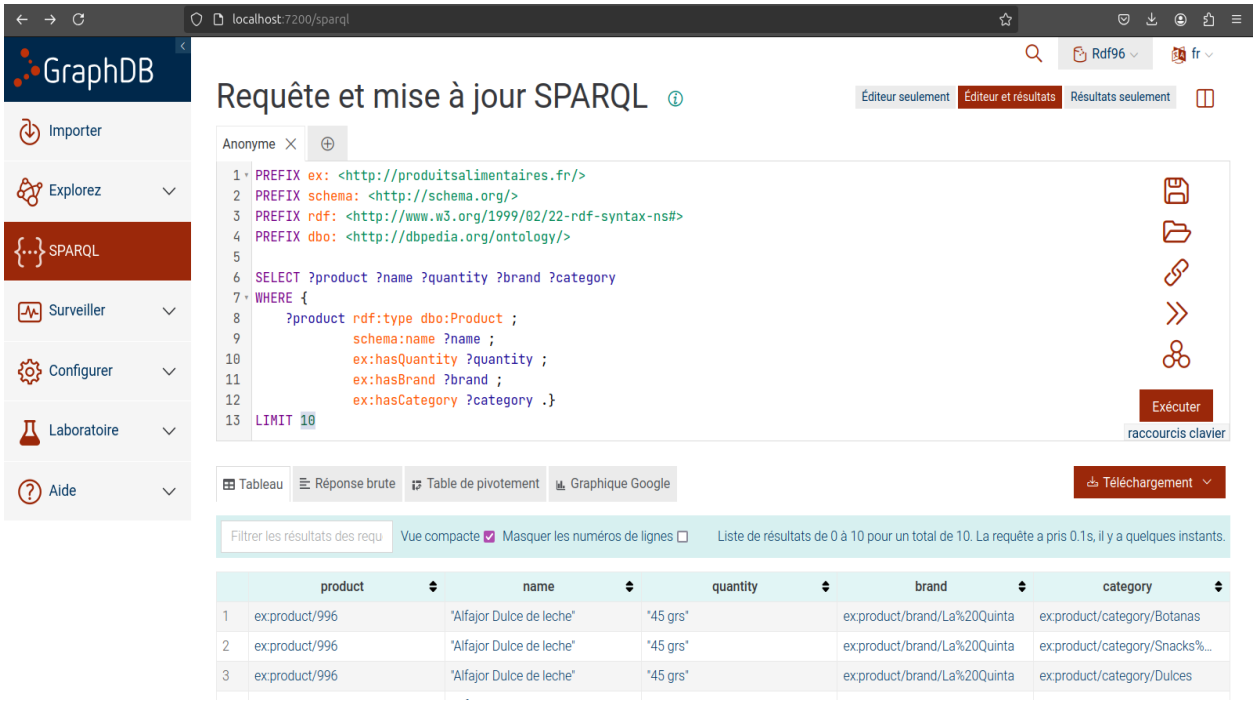
Aperçu de l'approche pour créer le graphe RDF en python:

```
import csv
import urllib.parse
import requests
from rdflib import Graph, Namespace, URIRef, Literal, RDF, RDFS
from rdflib.namespace import OWL, XSD, FOAF, DC
from SPARQLWrapper import SPARQLWrapper, JSON

# Chemins des fichiers
txt_file_path = "important_columns.txt"  # Fichier texte contenant les colonnes importantes
rdf_file_path = "output_ontology2.rdf"  # Fichier de sortie OWL

# Espaces de noms pour l'ontologie
SCHEMA = Namespace("http://schema.org/")
DBO = Namespace("http://dbpedia.org/ontology/")
EX = Namespace("http://produitsalimentaires.fr/")
FOAF = Namespace("http://xmlns.com/foaf/0.1/")
XSD = Namespace("http://www.w3.org/2001/XMLSchema#")

# Charger la liste des colonnes importantes depuis le fichier texte
try:
    with open(txt_file_path, "r") as file:
        important_columns = [line.strip() for line in file.readlines() if line.strip()]
except Exception as e:
    print(f"Erreur lors du chargement des colonnes : {e}")
    exit()

# Liaison des namespaces au graphe
g = Graph()
g.bind("schema", SCHEMA)
g.bind("dbo", DBO)
g.bind("foaf", FOAF)
g.bind("dc", DC)
g.bind("ex", EX)
g.bind("owl", OWL)
```

```python
# Fonction pour encoder les URI de manière sécurisée
def encode_uri(uri):
    return urllib.parse.quote(uri, safe=":/#?&=~")

# Lire le fichier CSV et ajouter des instances
try:
    with open(csv_file_path, newline='', encoding='utf-8') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:

            if all(col in row for col in important_columns):
                product_uri = URIRef(encode_uri(f"http://produitsalimentaires.fr/product/{row['code']}"))

                # Ajouter le type de produit

                g.add((product_uri, RDF.type, DBO.Product))
                g.add((product_uri, SCHEMA.name, Literal(row["product_name"])))
                g.add((product_uri, SCHEMA.url, URIRef(row["url"])))



                # Ajouter la marque
                if row["brands"]:
                    brand_uri = URIRef(encode_uri(f"http://produitsalimentaires.fr/brand/{row['brands']}"))
                    g.add((product_uri, EX.hasBrand, brand_uri))
                    g.add((brand_uri, RDF.type, DBO.Brand))
                    g.add((brand_uri, FOAF.name, Literal(row["brands"], lang="en")))

                # Ajouter des catégories basées sur le champ 'categories'
                if row["categories"]:
                    categories = row["categories"].split(',')  # Séparer les catégories par des virgules
                    for category in categories:
                        category = category.strip()
                        if category:  # Vérifier que la catégorie n'est pas vide
                            category_uri = URIRef(encode_uri(f"http://produitsalimentaires.fr/category/{category}"))
```

6. Stockage

Pour le stockage et l'interrogation de nos jeux de données RDF, nous avons utilisé le triplestore GraphDB, une base de données RDF performante et conforme aux standards du web sémantique. Ce choix repose sur sa capacité à gérer des volumes importants de triplets, sa compatibilité avec SPARQL pour les requêtes, et son interface conviviale pour la gestion des données. Nos jeux de données RDF, préparés au format RDF (`.rdf`), ont été chargés dans le triplestore via l'interface utilisateur de GraphDB. Une fois les données chargées, elles sont immédiatement disponibles pour l'interrogation via SPARQL.



7. Interrogation des Données

Grâce à GraphDB, on a pu interroger efficacement nos données RDF tout en exploitant pleinement les relations sémantiques définies par notre ontologie. Chaque triplet RDF relie un sujet à un objet via une propriété. Pour vérifier la conformité de la construction de notre graphe, on a effectue les requêtes SPARQL :

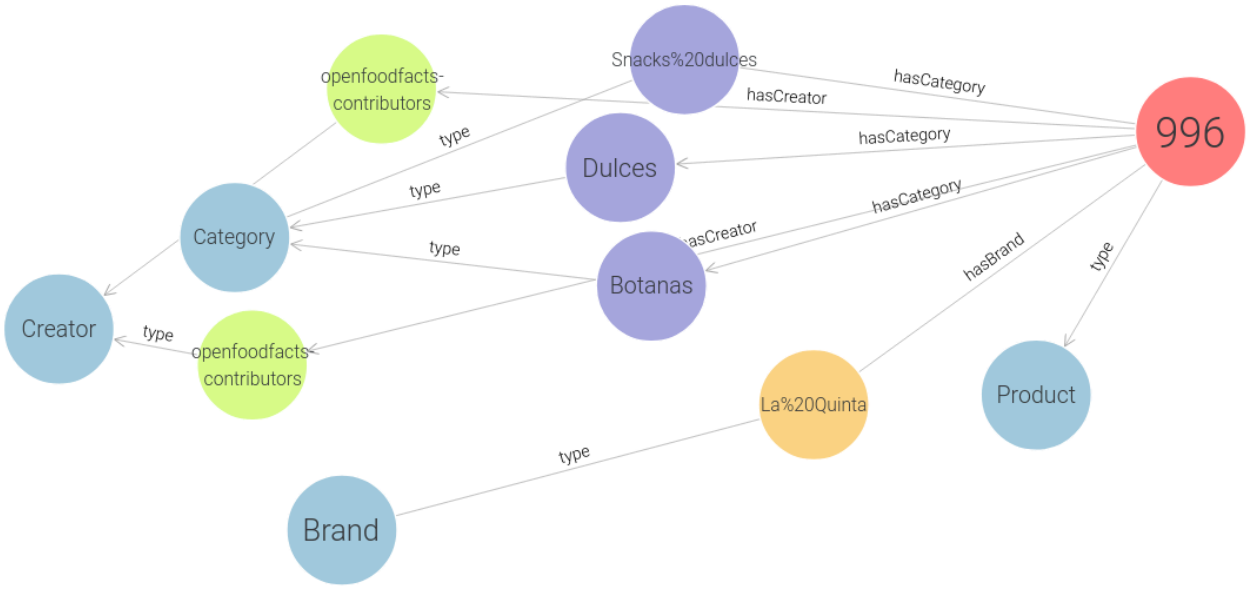
- Simples requêtes SPARQL
  - Trouver les informations détaillées sur 10 produits(leur code, leur nom, leur quantité et leur catégorie).

```
PREFIX ex: <http://produitsalimentaires.fr/>
PREFIX schema: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?product ?name ?quantity ?brand ?category
WHERE {
    ?product rdf:type dbo:Product ;
             schema:name ?name ;
             ex:hasQuantity ?quantity ;
             ex:hasBrand ?brand ;
             ex:hasCategory ?category .}
LIMIT 10
```

résultat

	product	name	quantity	brand	category
1	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:product/category/Bo
2	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:product/category/Sn
3	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:product/category/Du
4	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:product/category/Ali
5	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:product/category/Ga
6	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:product/category/Du
7	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:product/category/Alf
8	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:product/countries/A
9	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:category/Botanas
10	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	ex:product/brand/La%20Quinta	ex:category/Snacks%20

- Extraction d’un graphe RDF contenant les informations sur des produits.



- Description d'un produit spécifique

```
PREFIX ex: <http://produitsalimentaires.fr/>
PREFIX schema1: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

DESCRIBE ?product
WHERE {
   ?product rdf:type dbo:Product ;
            schema1:name "Alfajor Dulce de leche";
            ex:hasBrand ?brand .
  ?brand foaf:name ?brandName .
}
```

Résultat.

	subject	predicate	object
1	ex:product/996	rdf:type	dbo:Product
2	ex:product/996	schema:name	"Alfajor Dulce de leche"
3	ex:product/996	schema1:url	http://world-en.openfoodfacts.org/product/00000996...
4	ex:product/996	ex:hasBrand	ex:product/brand/La%20Quinta
5	ex:product/996	ex:hasBrand	ex:brand/La%20Quinta
6	ex:product/996	ex:hasCategory	ex:product/category/Botanas
7	ex:product/996	ex:hasCategory	ex:product/category/Snacks%20dulces
8	ex:product/996	ex:hasCategory	ex:product/category/Dulces
9	ex:product/996	ex:hasCategory	ex:product/category/Alimentos%20festivos
10	ex:product/996	ex:hasCategory	ex:product/category/Gastronom%C3%ADa%20navide...

- Vérification de l'existence d'un produit avec une marque spécifique

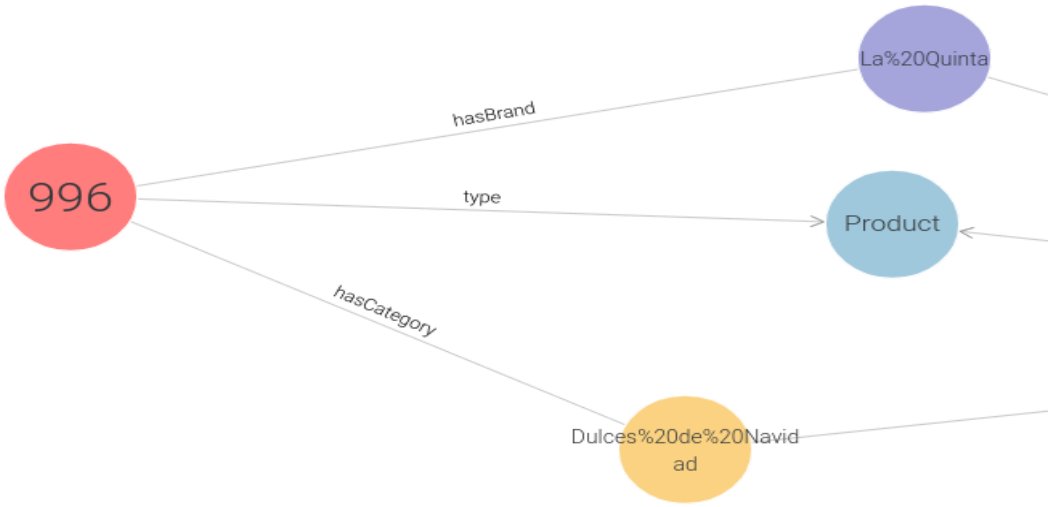
```
PREFIX ex: <http://produitsalimentaires.fr/>
PREFIX schema1: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK {
  ?product rdf:type dbo:Product ;
           schema1:name ?name ;
           ex:hasBrand ?brand .
  ?brand foaf:name ?brandName .
  # Utilisation de LCASE pour rendre la recherche insensible à la casse
  FILTER (LCASE(STR(?name)) = "alfajor dulce de leche" && LCASE(STR(?brandName)) = "la quinta") }
```

Résultat.

OUI

- Requêtes SPARQL complexes

- Construction d’un graphe RDF avec des informations détaillées sur les produits, leur marque et catégorie, tout en utilisant des filtres et des extractions sur la quantité(> 5 grs) pour obtenir des données plus structurées.



- Sélection des informations sur des produits, tout en utilisant des filtres (sur la quantité(>10) et la catégorie (=”dulces de navidad”)) pour ne sélectionner que les produits répondant aux critères définis .

```
PREFIX ex: <http://produitsalimentaires.fr/>
PREFIX schema: <http://schema.org/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?product ?name ?quantity ?numericQuantity ?brandName ?categoryName
WHERE {
  # Sélectionner les produits de type dbo:Product avec un nom spécifique
  ?product rdf:type dbo:Product ;
           schema:name ?name ;
           ex:hasCategory ?category ;
           ex:hasQuantity ?quantity ;
           ex:hasBrand ?brand .

  # Extraire la partie numérique de quantity en utilisant REGEX et la convertir en nombre
  BIND (xsd:decimal(REPLACE(?quantity, "[^0-9\\.]", "")) AS ?numericQuantity)

  # Associer chaque produit à une marque et une catégorie
  ?brand foaf:name ?brandName .
  ?category foaf:name ?categoryName .

  # Filtrer les produits dont la quantité est supérieure à 10 et qui appartiennent à la catégorie 'Dulces de Navidad'
  FILTER (?numericQuantity > 10 && LCASE(STR(?categoryName)) = "dulces de navidad"
}

ORDER BY ?name
```

Résultat:

	product	name	quantity	numericQuantity	brandName	categoryName
1	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	"45"^^xsd:decimal	"La Quinta"@en	"Dulces de Navidad"@en
2	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	"45"^^xsd:decimal	"La Quinta"@en	"Dulces de Navidad"@en
3	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	"45"^^xsd:decimal	"La Quinta"@en	"Dulces de Navidad"@en
4	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	"45"^^xsd:decimal	"La Quinta"@en	"Dulces de Navidad"@en
5	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	"45"^^xsd:decimal	"La Quinta"	"Dulces de Navidad"@en
6	ex:product/996	"Alfajor Dulce de leche"	"45 grs"	"45"^^xsd:decimal	"La Quinta"	"Dulces de Navidad"@en

8. Enrichissement des Données

Pour l'enrichissement de notre base de connaissances, nous avons associé nos données à des entités issues de **DBpedia** et **Wikidata**. Ces associations permettent de relier nos données à d'autres jeux de données disponibles sur le web sémantique, renforçant ainsi l'interopérabilité et l'accès à des informations enrichies et contextualisées.

Concrètement, les étapes d'intégration ont inclus :

- **Association des noms de pays** : Chaque pays a été lié à son URI correspondant sur Wikidata, en particulier à l'entité représentant son **QCode**, facilitant ainsi son identification unique dans le graphe.
- **Association des noms de marques** : Les noms de marques présents dans nos données ont été reliés aux URI correspondantes dans DBpedia, permettant d'accéder à des informations supplémentaires telles que l'historique, les fondateurs ou encore les catégories associées à chaque marque.
- **Association des catégories** : Les catégories de produits ont été associées à leur **QCode** respectif sur Wikidata. Cela permet de relier chaque catégorie à une entité unique et standardisée, ouvrant l'accès à des descriptions détaillées et à des relations hiérarchiques ou transversales définies dans Wikidata.

Ces liens sémantiques ouvrent la voie à une interopérabilité accrue et à une exploitation plus riche de notre graphe de connaissances au sein de l'écosystème du web des données.

9. Interrogation des données liées (Linked Data)

Pour exploiter les liens avec des ressources externes associés dans notre graphe de connaissance pour son enrichissement, nous avons effectué certaines requêtes fédérées:

- trouver le qcode de chaque categorie

```
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?categoryURI ?wikidataQCode
WHERE {
  ?categoryURI dbo:wikipage ?wikidataQCode .
```

}

Insérer une capture du résultat.

- Trouver pour un marque specifie , le nom du produit; le code; le qcode de la marque, le qcode et du nom du pays de la marque, le qcode et la categorie associe a la marque

```
 PREFIX ex: <http://example.org/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?productName ?productCode ?brandQCode ?brandCountryName
?brandCountryQCode ?categoryName ?categoryQCode
WHERE {
  # Étape 1 : Récupérer les informations sur la marque (QCode et URI) depuis
vos données locales
  ?brand a ex:Brand ;
        ex:hasName ?brandName ;
        dbo:wikipage ?brandQCode .
  FILTER (lcase(str(?brandName)) = lcase("nom_de_la_marque")) # Remplacez
par le nom de la marque

  # Étape 2 : Récupérer les produits associés à la marque
  ?product a ex:Product ;
        ex:hasName ?productName ;
        ex:hasCode ?productCode ;
        ex:hasBrand ?brand .

  # Étape 3 : Récupérer les informations sur le pays associé à la marque
  ?brand ex:hasCountry ?brandCountry .
  ?brandCountry dbo:wikipage ?brandCountryQCode ;
                ex:hasName ?brandCountryName .
```

```
  # Étape 4 : Récupérer les informations sur la catégorie associée à la marque
  ?brand ex:hasCategory ?category .
  ?category dbo:wikipage ?categoryQCode ;
        ex:hasName ?categoryName .
}
```

Insérer une capture du résultat.

- construction un sous-graphe avec des informations sur les produits, les marques, les pays associés, et les catégories

```
PREFIX ex: <http://example.org/>
PREFIX dbo: <http://dbpedia.org/ontology/>

CONSTRUCT {
  ?product ex:hasName ?productName ;
        ex:hasCode ?productCode ;
        ex:hasBrand ?brand .

  ?brand ex:hasName ?brandName ;
        dbo:wikipage ?brandQCode ;
        ex:hasCountry ?brandCountry ;
        ex:hasCategory ?category .

  ?brandCountry ex:hasName ?brandCountryName ;
              dbo:wikipage ?brandCountryQCode .

  ?category ex:hasName ?categoryName ;
        dbo:wikipage ?categoryQCode .
}
WHERE {
  # Étape 1 : Filtrer par le nom de la marque
```

```
  ?brand a ex:Brand ;
        ex:hasName ?brandName ;
        dbo:wikipage ?brandQCode .
  FILTER (lcase(str(?brandName)) = lcase("nom_de_la_marque")) # Remplacez
par le nom de la marque

  # Étape 2 : Associer les produits à la marque
  ?product a ex:Product ;
        ex:hasName ?productName ;
        ex:hasCode ?productCode ;
        ex:hasBrand ?brand .

  # Étape 3 : Récupérer le pays associé à la marque
  ?brand ex:hasCountry ?brandCountry .
  ?brandCountry dbo:wikipage ?brandCountryQCode ;
                ex:hasName ?brandCountryName .

  # Étape 4 : Récupérer la catégorie associée à la marque
  ?brand ex:hasCategory ?category .
  ?category dbo:wikipage ?categoryQCode ;
        ex:hasName ?categoryName .
}
```

Insérer une capture du résultat.

- requete avec les nutriments

```
PREFIX ex: <http://example.org/>
PREFIX dbo: <http://dbpedia.org/ontology/>

CONSTRUCT {
  ?product ex:hasName ?productName ;
        ex:hasCode ?productCode ;
```

```
        ex:hasBrand ?brand ;
        ex:hasCarbonFootprint ?carbonFootprint ;
        ex:hasFiber ?fiber ;
        ex:hasSugar ?sugar .

  ?brand ex:hasName ?brandName ;
        dbo:wikipage ?brandQCode ;
        ex:hasCountry ?brandCountry ;
        ex:hasCategory ?category .

  ?brandCountry ex:hasName ?brandCountryName ;
                dbo:wikipage ?brandCountryQCode .

  ?category ex:hasName ?categoryName ;
        dbo:wikipage ?categoryQCode .

  # Nutriments
  ?product ex:hasNutriment ?carbonFootprintNutriment .
  ?carbonFootprintNutriment ex:hasValue ?carbonFootprint .
  ?carbonFootprintNutriment ex:hasName "Carbon Footprint" .

  ?product ex:hasNutriment ?fiberNutriment .
  ?fiberNutriment ex:hasValue ?fiber .
  ?fiberNutriment ex:hasName "Fiber" .

  ?product ex:hasNutriment ?sugarNutriment .
  ?sugarNutriment ex:hasValue ?sugar .
  ?sugarNutriment ex:hasName "Sugar" .
}
WHERE {
  # Étape 1 : Filtrer par le nom de la marque
  ?brand a ex:Brand ;
```

```
        ex:hasName ?brandName ;
        dbo:wikipage ?brandQCode .
  FILTER (lcase(str(?brandName)) = lcase("nom_de_la_marque")) # Remplacez
par le nom de la marque

  # Étape 2 : Associer les produits à la marque
  ?product a ex:Product ;
        ex:hasName ?productName ;
        ex:hasCode ?productCode ;
        ex:hasBrand ?brand ;
        ex:hasNutriment ?carbonFootprintNutriment ;
        ex:hasNutriment ?fiberNutriment ;
        ex:hasNutriment ?sugarNutriment .

  # Étape 3 : Récupérer le pays associé à la marque
  ?brand ex:hasCountry ?brandCountry .
  ?brandCountry dbo:wikipage ?brandCountryQCode ;
                ex:hasName ?brandCountryName .

  # Étape 4 : Récupérer la catégorie associée à la marque
  ?brand ex:hasCategory ?category .
  ?category dbo:wikipage ?categoryQCode ;
        ex:hasName ?categoryName .

  # Nutriments (empreinte carbone, fibre, sucre)
  ?carbonFootprintNutriment ex:hasValue ?carbonFootprint .
  ?fiberNutriment ex:hasValue ?fiber .
  ?sugarNutriment ex:hasValue ?sugar .
}
```

Insérer une capture du résultat.

- vérifie si un produit spécifique possède certains nutriments, en particulier l'empreinte carbone, les fibres et le sucre, ainsi que d'autres informations liées comme la marque, le pays et la catégorie de ce produit.

```
PREFIX ex: <http://example.org/>
PREFIX dbo: <http://dbpedia.org/ontology/>

ASK {
 # Vérification de l'existence d'un produit avec un nom spécifique
 ?product a ex:Product ;
        ex:hasName "Product A" ;
        ex:hasCode ?productCode ;
        ex:hasBrand ?brand ;
        ex:hasNutriment ?carbonFootprintNutriment ;
        ex:hasNutriment ?fiberNutriment ;
        ex:hasNutriment ?sugarNutriment .

 # Vérification des nutriments associés
 ?carbonFootprintNutriment ex:hasValue ?carbonFootprint .
 ?fiberNutriment ex:hasValue ?fiber .
 ?sugarNutriment ex:hasValue ?sugar .

 # Vérification des informations de la marque
 ?brand ex:hasName ?brandName ;
        dbo:wikipage ?brandQCode ;
        ex:hasCountry ?brandCountry ;
        ex:hasCategory ?category .

 # Vérification du pays de la marque
 ?brandCountry ex:hasName ?brandCountryName ;
                dbo:wikipage ?brandCountryQCode .
```

```
  # Vérification de la catégorie de la marque
  ?category ex:hasName ?categoryName ;
        dbo:wikipage ?categoryQCode .
}
```

Insérer une capture du résultat.

- Obtention de toutes les informations relatives à un produit spécifique (par exemple, "Product A"), ainsi que des informations sur la marque, le pays, la catégorie et les nutriments associés.

```
PREFIX ex: <http://example.org/>
PREFIX dbo: <http://dbpedia.org/ontology/>

DESCRIBE ?product ?brand ?brandCountry ?category
?carbonFootprintNutriment ?fiberNutriment ?sugarNutriment {
  # Récupérer le produit spécifique
  ?product a ex:Product ;
        ex:hasName "Product A" ;   # Nom du produit, à remplacer par le nom
exact
        ex:hasCode ?productCode ;
        ex:hasBrand ?brand ;
        ex:hasNutriment ?carbonFootprintNutriment ;
        ex:hasNutriment ?fiberNutriment ;
        ex:hasNutriment ?sugarNutriment .

  # Nutriments associés
  ?carbonFootprintNutriment ex:hasValue ?carbonFootprint ;
                ex:hasName "Carbon Footprint" .
  ?fiberNutriment ex:hasValue ?fiber ;
        ex:hasName "Fiber" .
```

```
  ?sugarNutriment ex:hasValue ?sugar ;
                  ex:hasName "Sugar" .

  # Marque associée au produit
  ?brand ex:hasName ?brandName ;
         dbo:wikipage ?brandQCode ;
         ex:hasCountry ?brandCountry ;
         ex:hasCategory ?category .

  # Pays associé à la marque
  ?brandCountry ex:hasName ?brandCountryName ;
                dbo:wikipage ?brandCountryQCode .

  # Catégorie de la marque
  ?category ex:hasName ?categoryName ;
         dbo:wikipage ?categoryQCode .
}
```

Insérer une capture du résultat.

10. Outils et technologies utilisés

Pour la réalisation de notre projet, nous avons utilisé un ensemble varié de technologies et d'outils qui ont joué un rôle clé dans les différentes étapes de conception et de mise en œuvre. Parmi ces outils :

- **Protégé** : Un éditeur puissant pour la création et la gestion de notre ontologie, permettant de définir des classes, des propriétés et des relations sémantiques.
- **Triplestore** (**GraphDB**) : pour le stockage des données RDF et l'exécution des requêtes SPARQL, essentiels pour interroger et manipuler le graphe de connaissances.

- **Python** : utilisé pour automatiser plusieurs tâches, notamment le traitement des données, la conversion du fichier CSV en triplets RDF, facilitant ainsi leur intégration dans le graphe de connaissances(via des bibliothèques comme **rdflib**), la conversion du fichier RDF au format Turtle et JSON-LD, et l'interaction avec le triplestore (requêtes SPARQL via **SPARQLWrapper**),.
- **Sources de données liées** (telles que **DBpedia** et **Wikidata**) : Exploitées pour l'enrichissement de notre base de connaissances en intégrant des informations issues du web sémantique.