A.Wilk

AtsuhikoMochizuki

IBOOF

Immersive & Best Open fOod Facts advisor

Développement d'une application JAVA multi-couches basée sur la base de données open-source Open Food Facts©

digi namic FORMATION

Dossier de conception

Sous la direction de Séga Sylla

03/07/23

https://github.com/atsuhikoMochizuki/IBOOF.git

Cahier des charges

Description

Vous connaissez peut-être l'application Yuka, disponible sur smartphone. Yuka fournit des informations nutritionnelles sur pratiquement tous les produits alimentaires commercialisés en France.

En plus d'informations, elle fournit également un score nutritionnel, de A (excellent) à F (mauvais).

Cette application à succès s'est construite sur une base de données open source appelée Open Food Facts.

La base de données Open Food Facts est une base de données mondiale, qu'on peut télécharger sous la forme d'un fichier CSV. Le fichier que je vais vous demander de traiter dans le cadre de ce projet est le même que celui sur lequel s'est basé Yuka. Il ne concerne que les produits alimentaires fabriqués en France.

Dans ce TP vous allez créer une application qui met en base ce fichier.

Description du contenu du fichier

Le fichier open food facts, au format CSV, comporte 13 432 références de produits avec 30 informations associées par produit. Dans ce fichier le caractère séparateur est le |. Si on découpe selon ce caractère séparateur, on obtient le tableau suivant :

- Index 0 : catégorie du produit
- Index 1 : marque du produit
- Index 2 : nom du produit

- Index 3 : score nutritionnel : A (excellent) à F (mauvais)
- Index 4 : liste des ingrédients séparés la plupart du temps par des virgules, mais pas toujours.
- Index 5 : énergie pour 100g (en joules)
- Index 6 : quantité de graisse pour 100g

  etc.
- Index 28 : liste des allergènes séparés la plupart du temps par des virgules
- Index 29 : liste des additifs séparés la plupart du temps par des virgulesObjectifs

Le but est de concevoir et développer une application basée sur l'API JPA pour mettre en base toutes ces données.

Les règles de gestion à respecter :

- Une catégorie doit être unique en base de données
- Une marque doit être unique en base de données
- Un ingrédient doit être unique en base de données.
- Un allergène doit être unique en base de données
- Un additif doit être unique en base de données
- Cas particulier du traitement des ingrédients :
  - Dans le fichier, les ingrédients sont séparés par un séparateur.

    Exemple : Sucre, farine, banane

Le séparateur le plus courant est la virgule mais il est possible que d'autres séparateurs aient été utilisés comme le tiret. En effet comme il s'agit d'une base mondiale, certains utilisateurs ne respectent pas toujours les consignes.

Attention donc à prévoir plusieurs cas de figures:

- De manière à éviter les ingrédients du type Sucre* ou _Sucre_ par exemple, il faudra prévoir une suppression des caractères parasites.
- Certains ingrédients possèdent une description entre parenthèses : il est demande de supprimer toutes les informations présentes entre parenthèses.
- Certains ingrédients possèdent également des %. De manière à éviter d'avoir des ingrédients du type Sucre 50%, il faudra veiller à supprimer tous les pourcentages.
- Cas d'exemples pour les traitements des ingrédients, des allergènes et des additifs :
    - Cas 1 – les caractères spéciaux :
        - Avant traitement : Sucre*, farine, _Maïs_
        - Après traitement : Sucre, farine, Maïs
    - Cas 2 – les pourcentages :
        - Avant traitement : Sucre 15%, farine 50%, Maïs 35%
        - Après traitement : Sucre, farine, Maïs
    - Cas 3 – les parenthèses :
        - Avant traitement : Sucre, banane, Pâte (Farine 50%, Sucre 20%, Œufs 30%)
        - Après traitement : Sucre, banane, Pâte

Objectif n°1 : Réaliser un dossier de conception

Ce dernier contiendra:

- Un diagramme de classes métier
- Un modèle physique de données

Prévoyez 1 journée pour ce travail sinon vous n'aurez pas le temps de réaliser le développement. Vous placerez le dossier de conception dans un répertoire appelé conception et situé à la racine de votre projet Git.

Objectif n°2 : Développer l'application

- Mettre en en place les entités JPA avec les annotations
- Développer de manière professionnelle en tenant compte des bonnes pratiques de codage : javadoc, pas de code redondant, etc.
- Mettez en place une couche DAO, acronyme de Data Access Object, (une DAO par entité) pour les accès à la base de données.

  Si vous voulez en savoir plus sur le pattern DAO : https://www.baeldung.com/java-dao-pattern voir notamment le chapitre 3.1.

Objectif n°3 : mettre en place des optimisations afin que votre traitement soit le plus rapide possible.

Vous pourrez essayer par exemple de mettre en place le cache.

Record à battre : 35 minutes en respectant les règles de gestion bien sûr.

Objectif n°4 : Si vous avez terminé l'objectif 3, développez une application de restitution avec un menu et permettant :

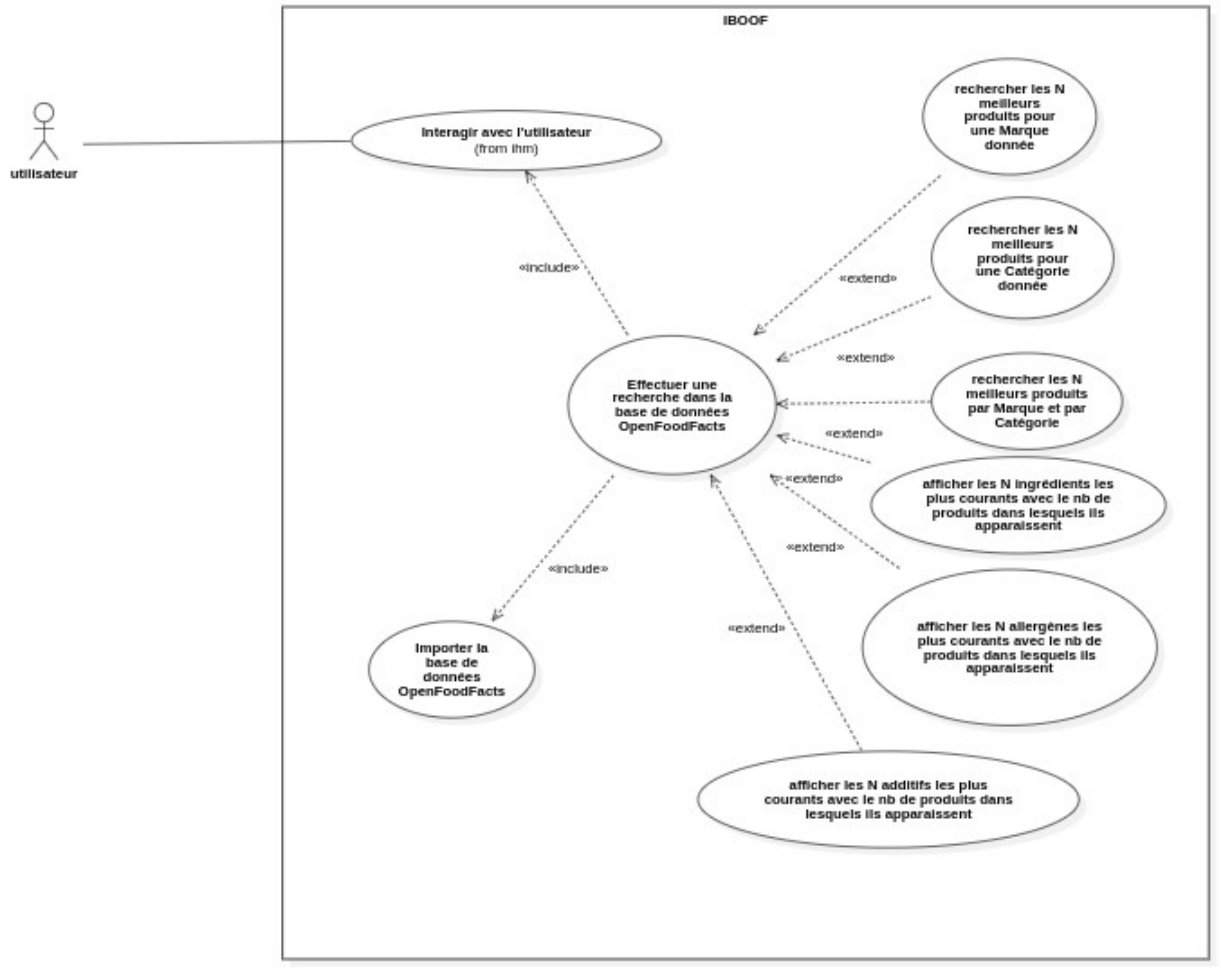
- de rechercher les N meilleurs produits pour une Marque donnée. La marque et la valeur de N sont demandées à l'utilisateur.
- de rechercher les N meilleurs produits pour une Catégorie donnée. La catégorie et la valeur de N sont demandées à l'utilisateur.
- de rechercher les N meilleurs produits par Marque et par Catégorie. La marque, la catégorie et la valeur de N sont demandées à l'utilisateur.
- d'afficher les N ingrédients les plus courants avec le nb de produits dans lesquels ils apparaissent. La valeur de N est demandée à l'utilisateur.
- d'afficher les N allergènes les plus courants avec le nb de produits dans lesquels ils apparaissent. La valeur de N est demandée à l'utilisateur.
- d'afficher les N additifs les plus courants avec le nb de produits dans lesquels ils apparaissent. La valeur de N est demandée à l'utilisateur.

Table des matières

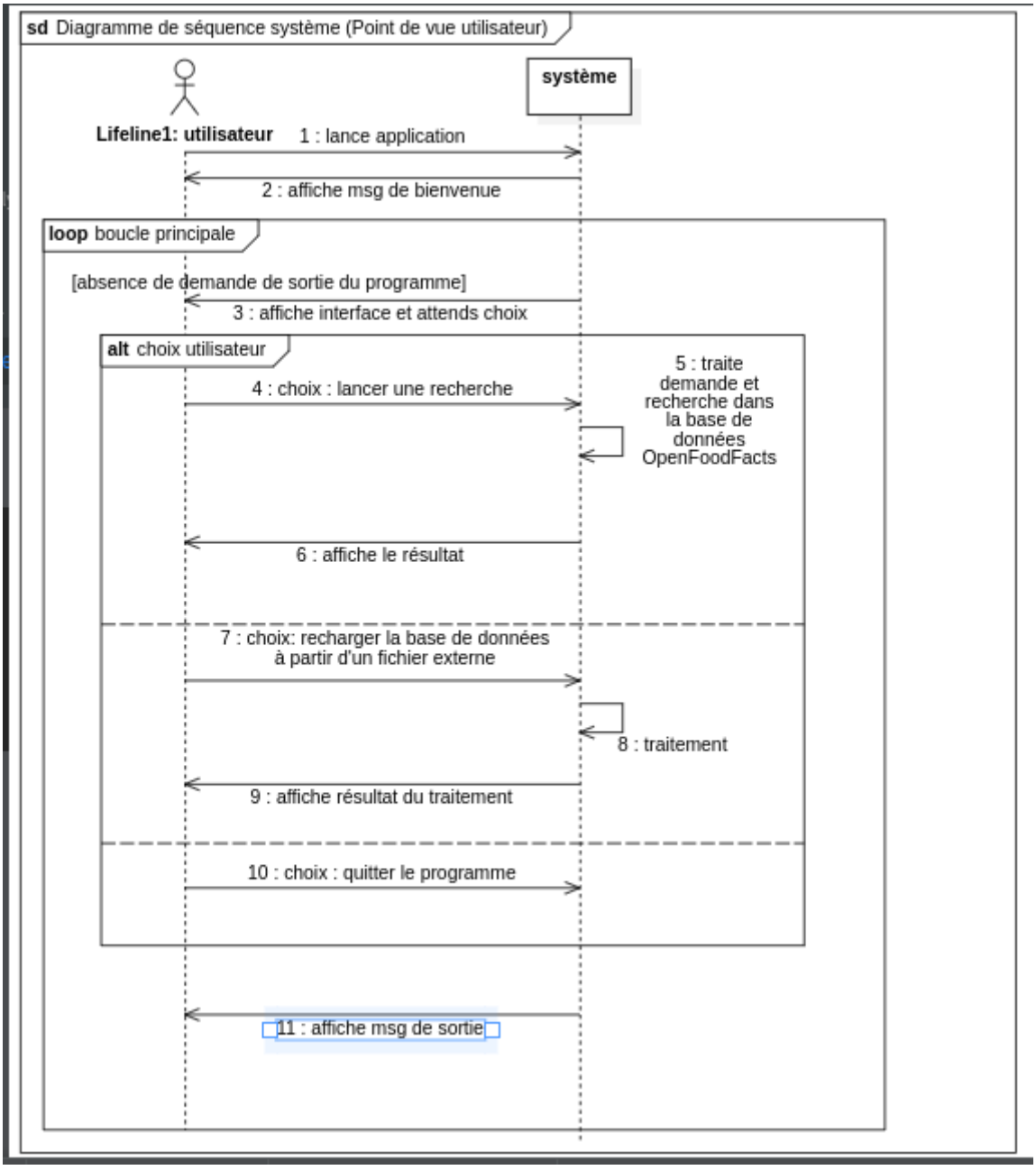
1. Analyse

1.1. Diagramme des cas d'utilisation

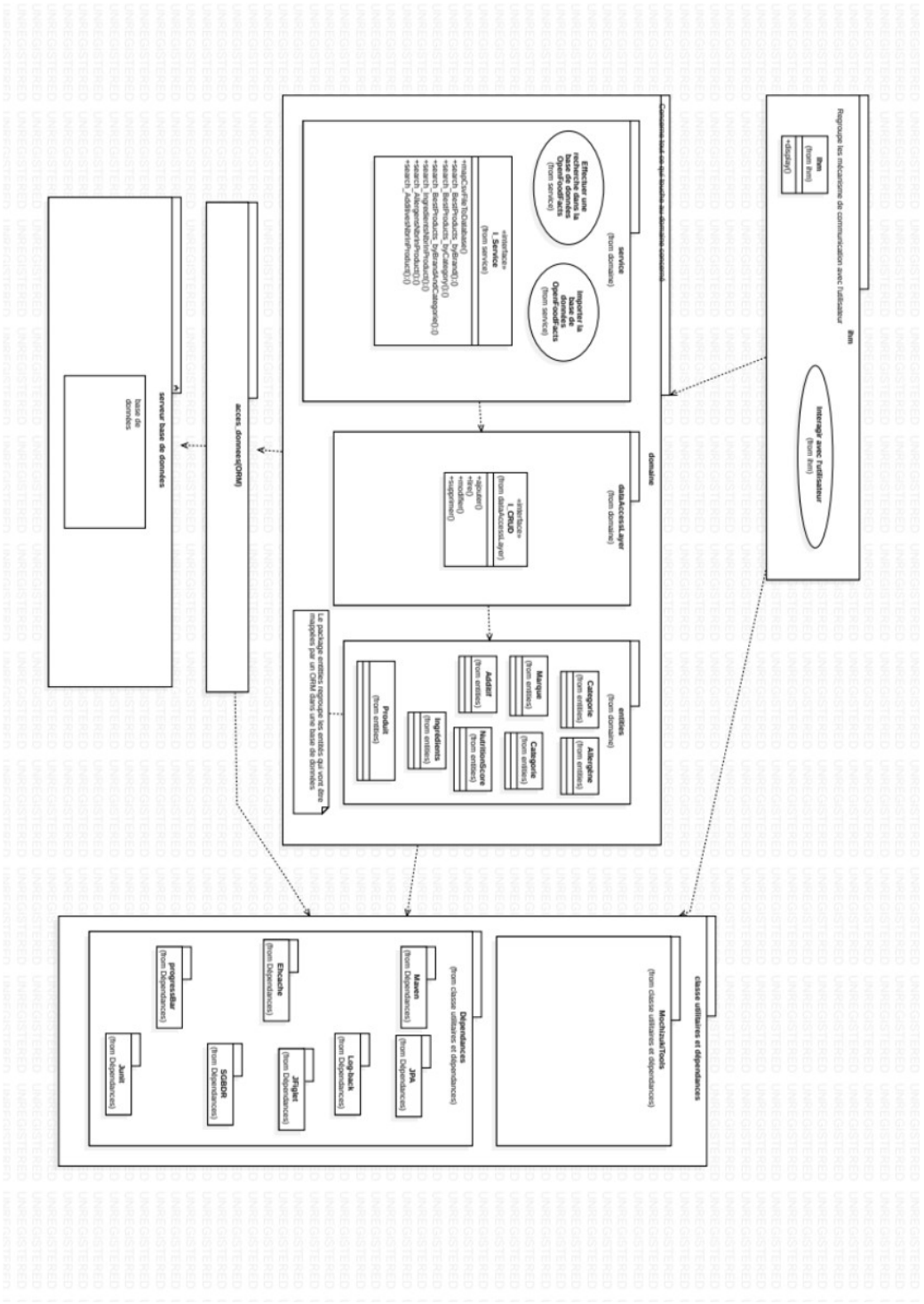
Afin de mieux appréhender le système à concevoir, nous commençons par représenter les fonctionnalités requises par l'application. Nous adoptons pour cela un point de utilisateur, au travers un diagramme de cas d'utilisation.



1.2. Comportement de l'application d'un point de vue utilisateur (diagramme de séquence système)



1.3. Répartition de la responsabilité des fonctionnalités au sein de différentes couches et modélisation des entités mises en jeu pour les réaliser



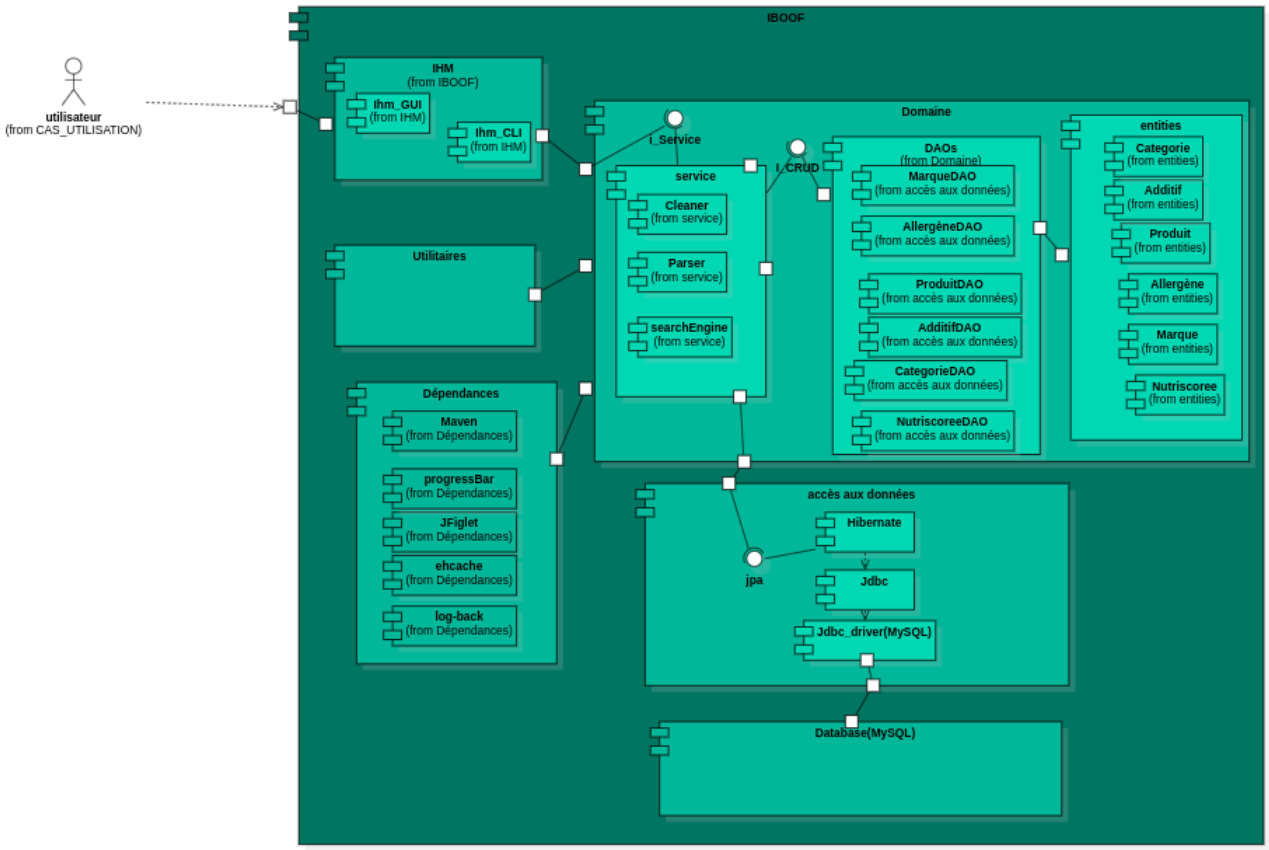
2. Conception de l’application multi-couche

Dans un soucis de modularité, notre application sera structurée en différentes couches.

Les différents paquetages dans lequels nous avons regroupés nos classes, seront à présent représentés, dans un point de vue conceptuel, sous forme composants.

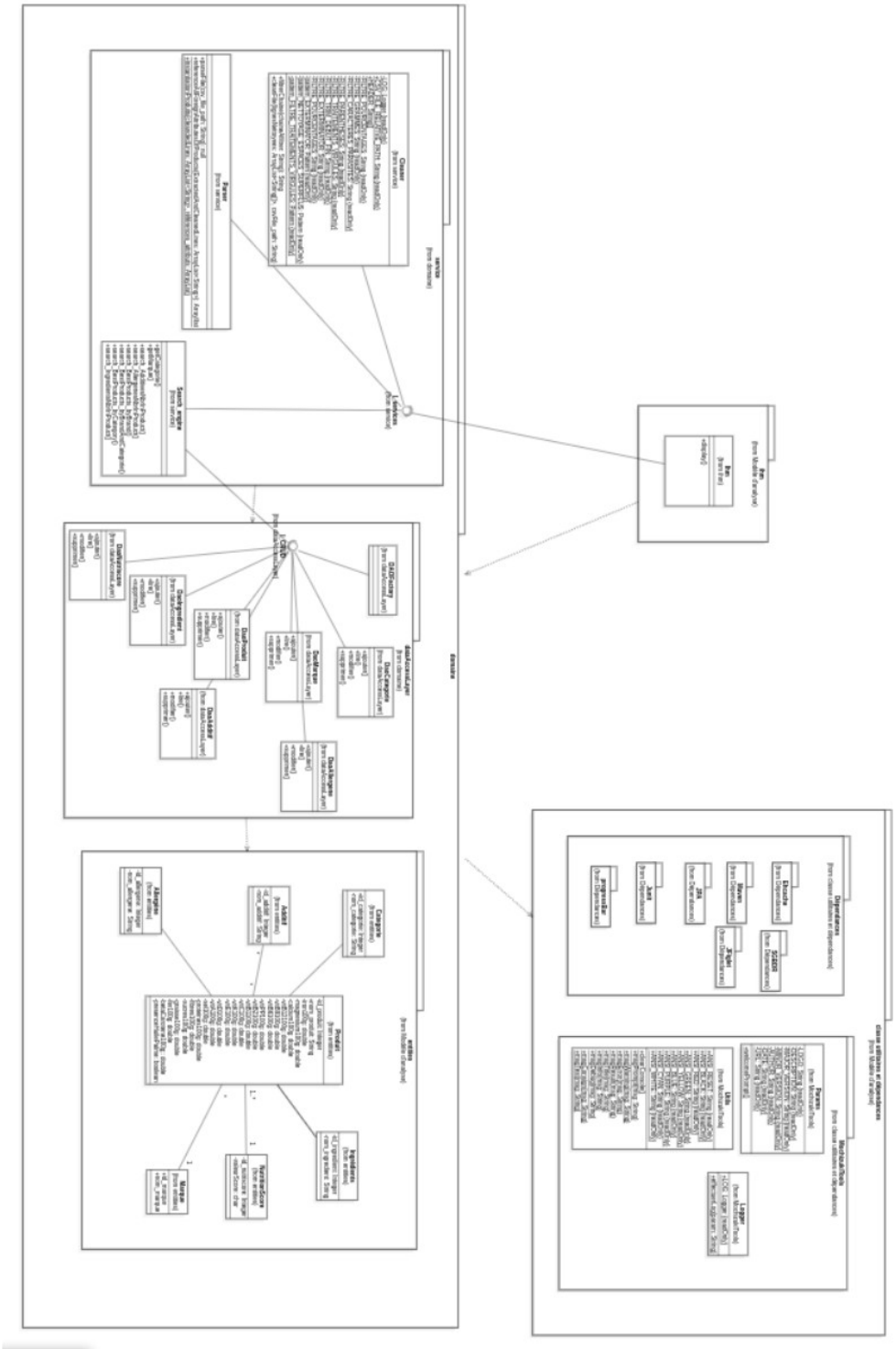
2.1. Vue générale du système

Au vu de ce qui a pu émerger lors de notre phase d'analyse et des regroupement des différentes entités et responsabilités, l'application sera implémentée selon les couches suivantes :


IBOOF
utilisateur
(from CAS_UTILISATION)
IHM
(from IBOOF)
Ihm_GUI
(from IHM)
Ihm_CLI
(from IHM)
Utilitaires
Dépendances
Maven
(from Dépendances)
progressBar
(from Dépendances)
JFiglet
(from Dépendances)
ehcache
(from Dépendances)
log-back
(from Dépendances)
Domaine
I_Service
I_CRUD
service
Cleaner
(from service)
Parser
(from service)
searchEngine
(from service)
DAOs
(from Domaine)
MarqueDAO
(from accès aux données)
AllergèneDAO
(from accès aux données)
ProduitDAO
(from accès aux données)
AdditifDAO
(from accès aux données)
CategorieDAO
(from accès aux données)
NutriscoreeDAO
(from accès aux données)
entities
Categorie
(from entities)
Additif
(from entities)
Produit
(from entities)
Allergène
(from entities)
Marque
(from entities)
Nutriscoree
(from entities)
accès aux données
jpa
Hibernate
Jdbc
Jdbc_driver(MySQL)
Database(MySQL)

2.2. Composants du système

2.2.1. Vue d'ensemble des classes



2.2.2. Interface Homme-machine

2.2.2.1. Choix technologiques

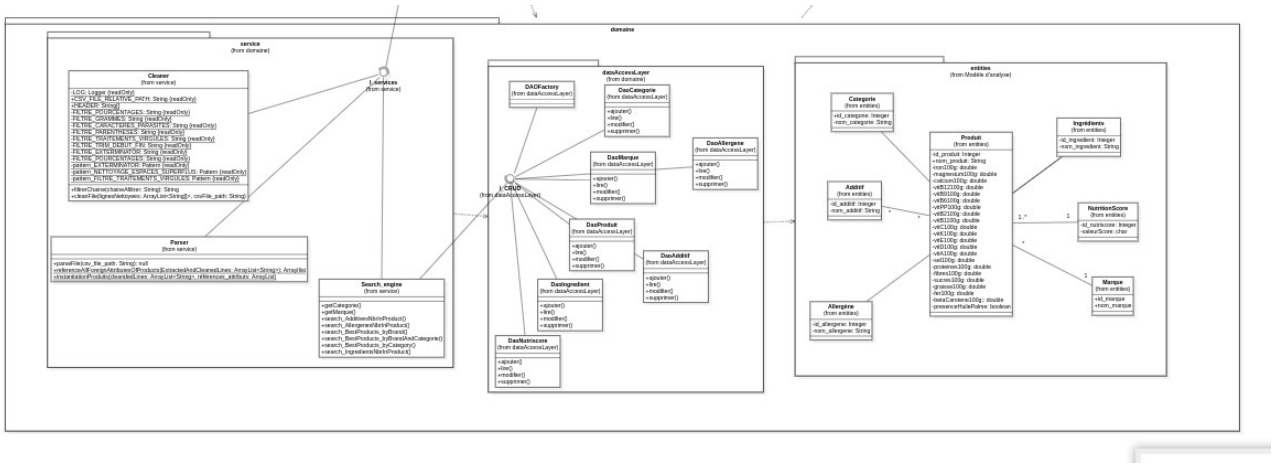
L'interfaçage avec l'utilisateur sera réalisé à l'aide d'une fenêtre graphique implémentée avec la libraire SWING.

Nous proposerons également une interface utilitaire de type console.

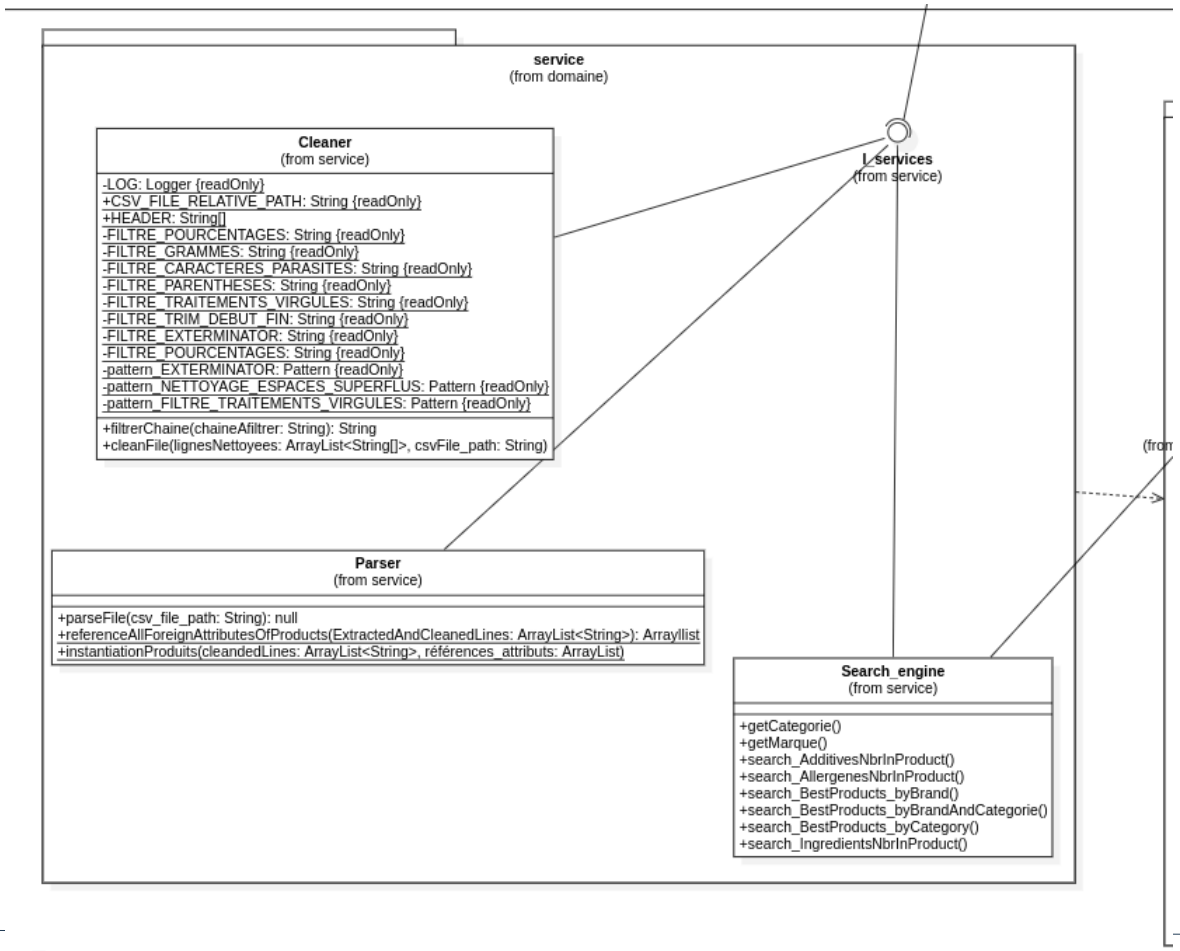
Maquette de l'interface

2.2.3. Composant domaine:

2.2.3.1. Vue d'ensemble

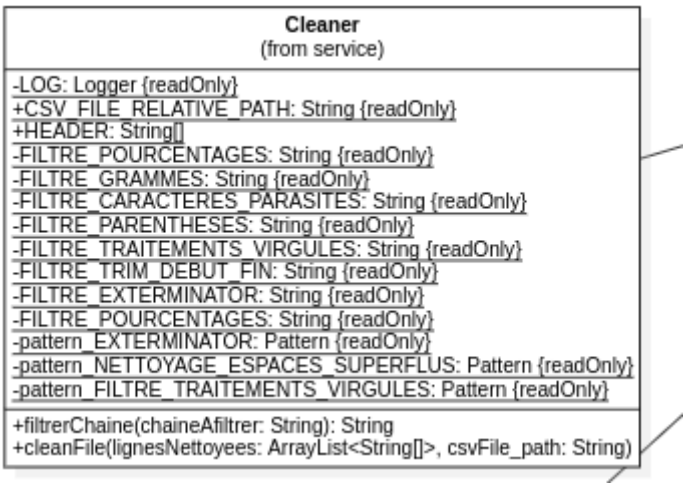


2.2.3.2. Service



2.2.3.2.1. Cleaner et filtres REGEX

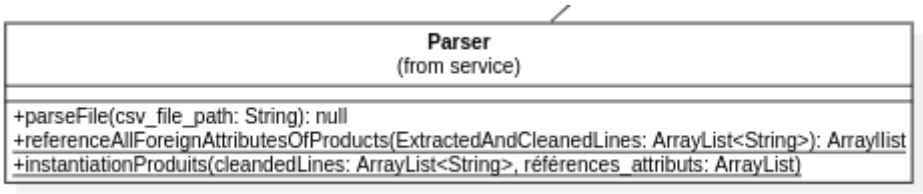
Le composant cleaner permet d'extraire les lignes brutes du fichier CSV de OpenFoodFacts et de les nettoyer



Les filtres de ce composant seront construits à l'aide d'expressions régulières (REGEX)

2.2.3.2.2. Parser

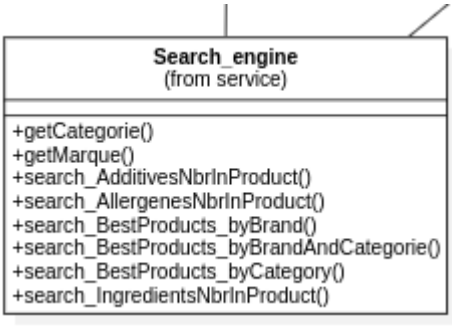
Le parser va permettre, en appellant au préalable le Cleaner, analyser et trier toutes les données(suppression des doublons, organisation des catégories….), puis va se charger de les insérer en base de données



2.2.3.2.3. Le moteur de recherche SearchEngine

Il sera en charge de faire les recherches en base de données en fonction des demandes en provenance de l'IHM.

Il dialoguera avec l'interface CRUD, proposée par la couche DAO

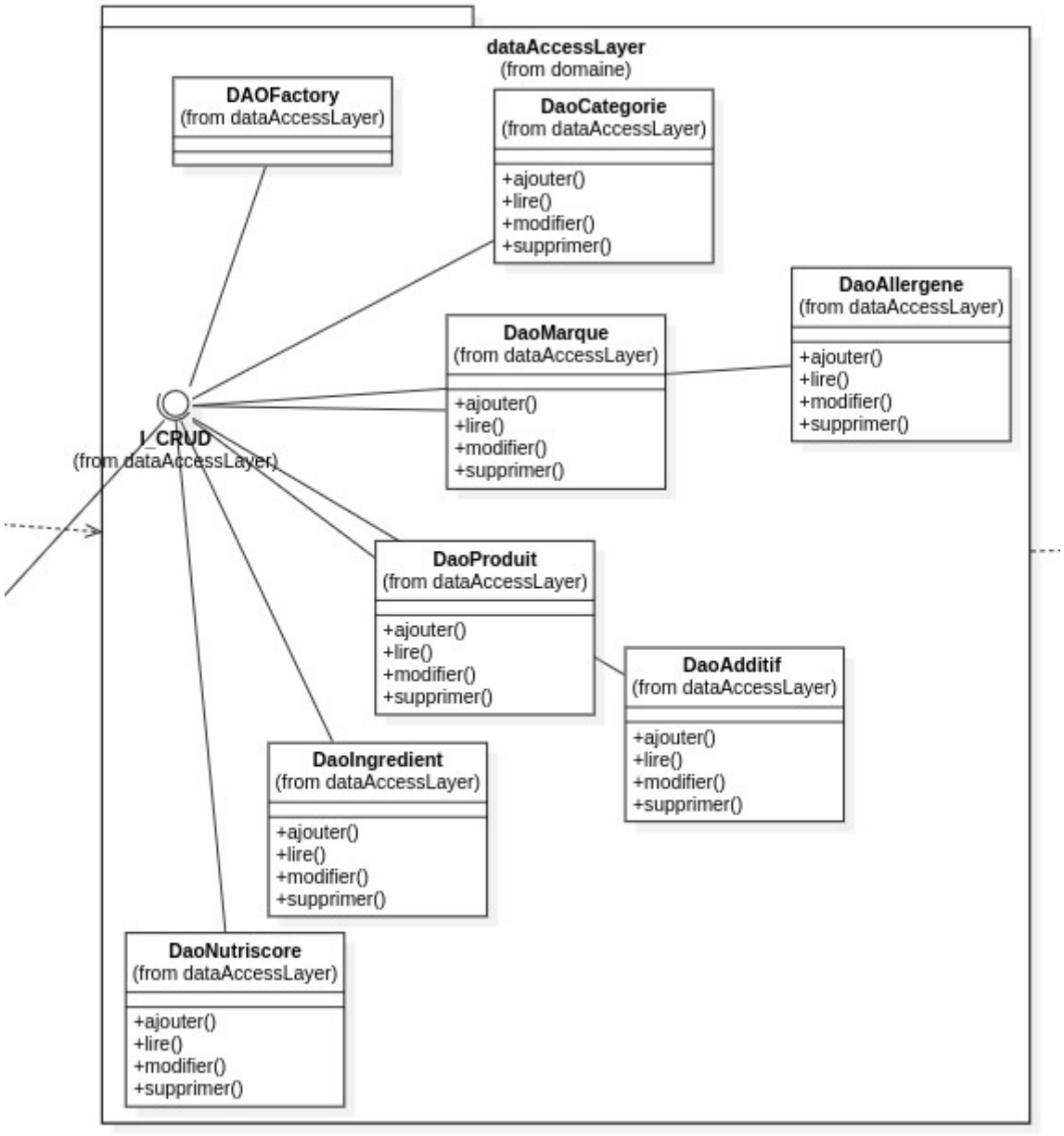


2.2.3.3. Data Access Layer

Ce composant s'expose à l'extérieur à l'aide d'une interface CRUD.

Le DAL est composé de DAO implémentant chacun le CRUD.

Tous son dédiées à une des entités de la couche entities. Par le biais du logiciel ORM, les entités seront quand à elles persistées en base de données.

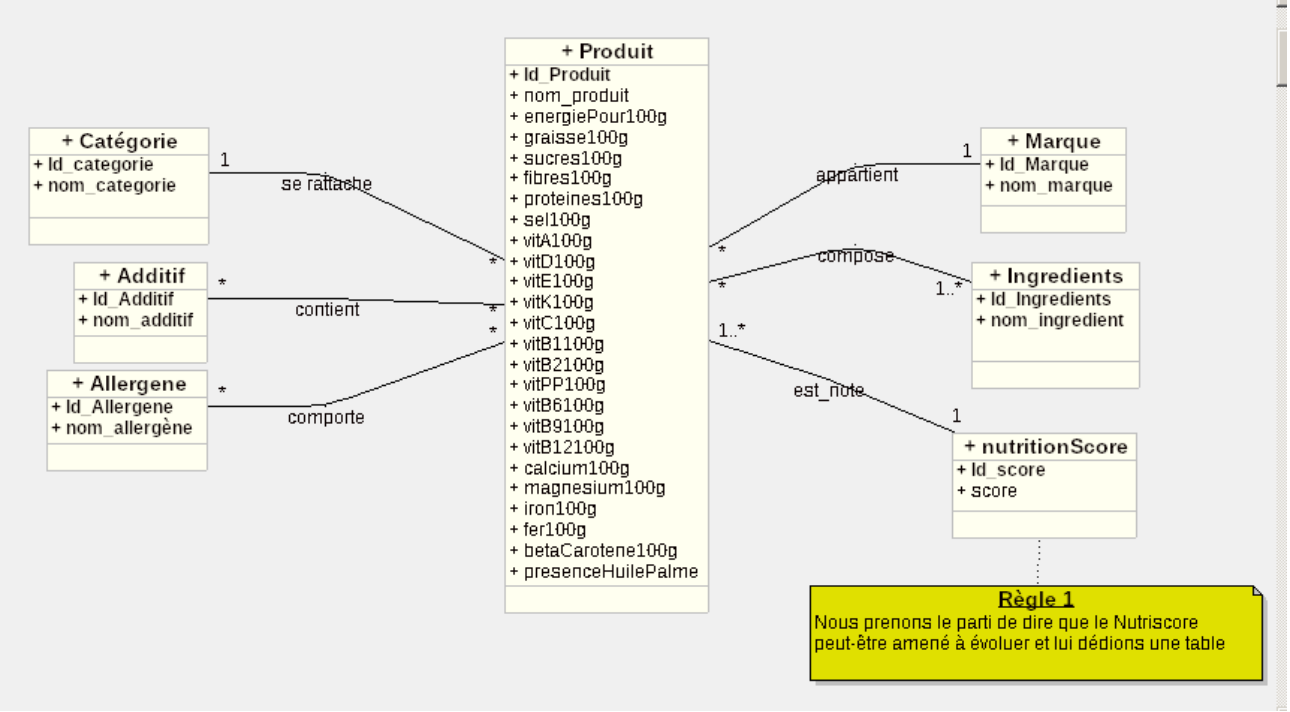


2.2.3.4. Entities

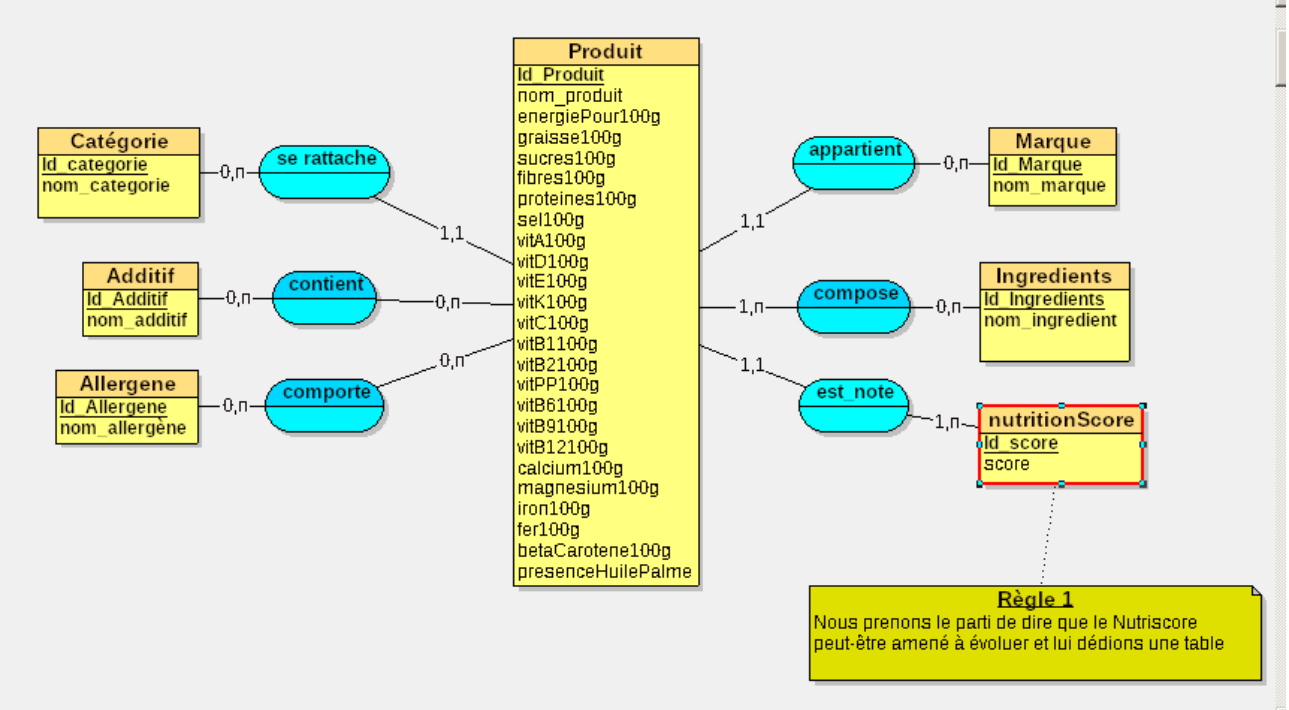
2.2.3.4.1. modèle conceptuel des données

Ce composants regroupes toutes les classes images des tables qui seront mappées via le logiciel ORM JPA.

Le diagramme des classes de ce package correspond au modèle conceptuel de données



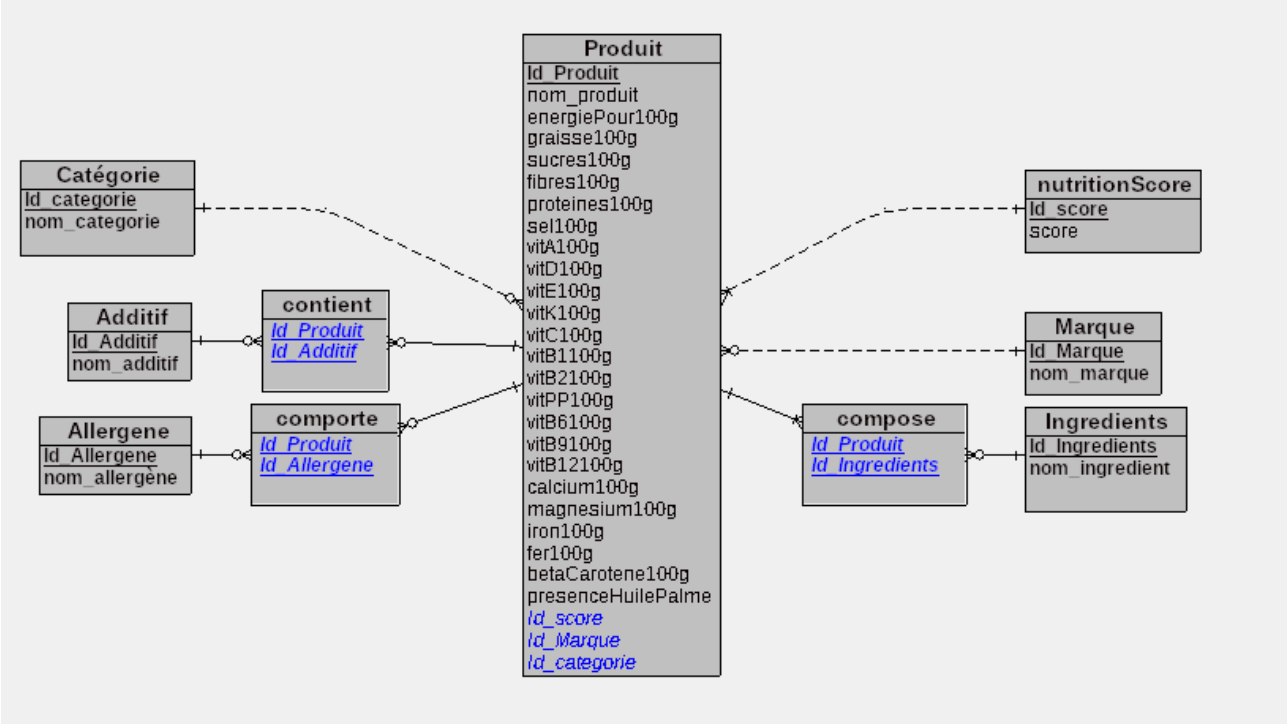
Le même diagramme modélisé selon la méthode Merise sous Looping :


Produit
Id_Produit
nom_produit
energiePour100g
graisse100g
sucres100g
fibres100g
proteines100g
sel100g
vitA100g
vitD100g
vitE100g
vitK100g
vitC100g
vitB1100g
vitB2100g
vitPP100g
vitB6100g
vitB9100g
vitB12100g
calcium100g
magnesium100g
iron100g
fer100g
betaCarotene100g
presenceHuilePalme
Catégorie
Id_categorie
nom_categorie
se rattache
0,n
1,1
Additif
Id_Additif
nom_additif
contient
0,n
0,n
Allergene
Id_Allergene
nom_allergène
comporte
0,n
0,n
appartient
1,1
0,n
Marque
Id_Marque
nom_marque
compose
1,n
0,n
Ingredients
Id_Ingredients
nom_ingredient
est_note
1,1
1,n
nutritionScore
Id_score
score
Règle 1
Nous prenons le parti de dire que le Nutriscore
peut-être amené à évoluer et lui dédions une table

2.2.3.4.2. Mapping vers le modèle physique des données(MPD)

Le modèle physique des données va représenter les données telles qu'elles vont être organisées en base.

Dans le modèle physique des données les relations vont être à l'aide de clés étrangères et de tables de jointures.

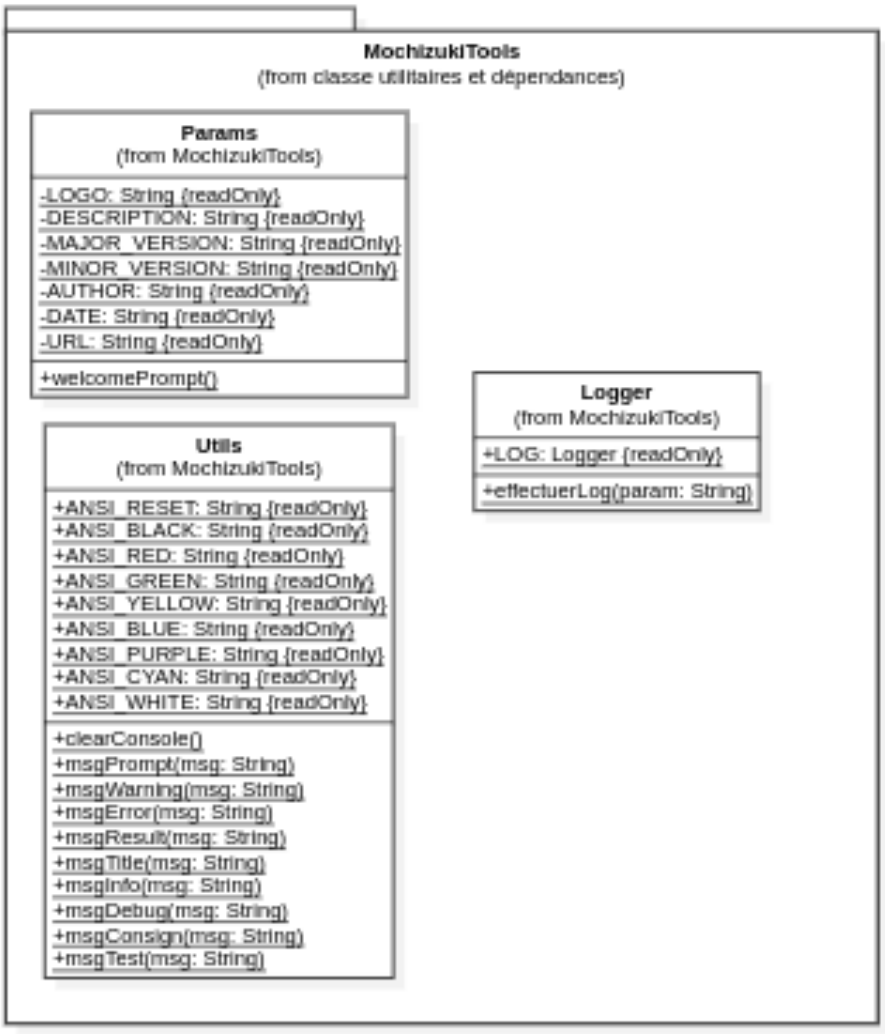


2.2.4. Utilitaires et dépendances

2.2.4.1. Utilitaires

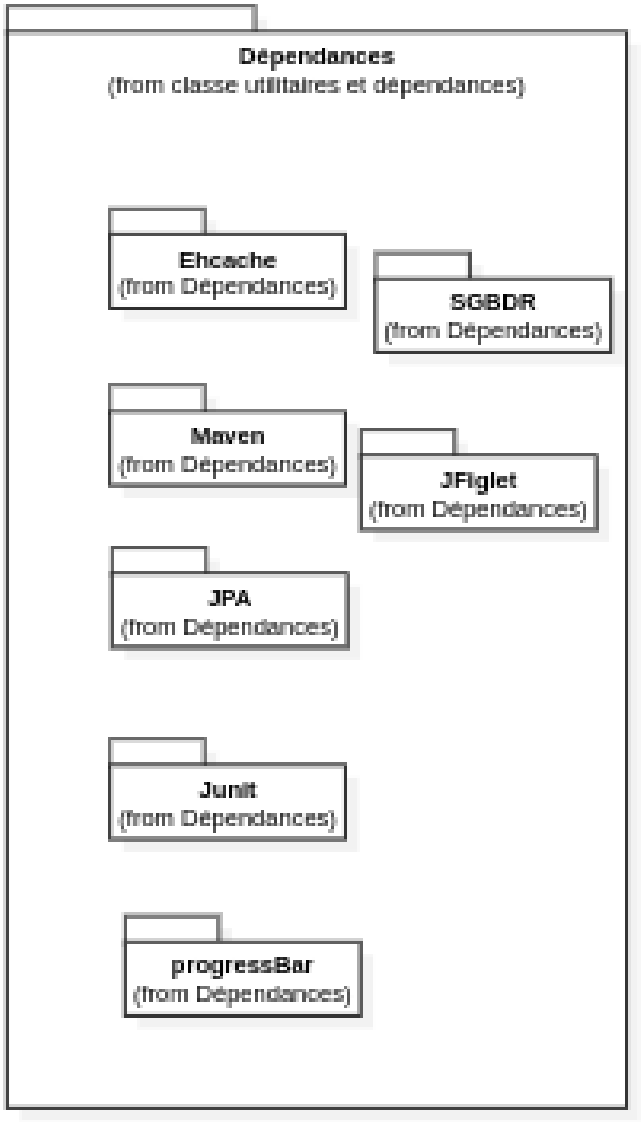
Ce composant va regrouper différentes classe utilitaires que nous allons implémenter :

- Logger : gestion des logs
- Params : Récupération des propriétés décrites dans le fichiers properties du projet, génération automatique d'en-tête programme.
- Utils : Affichage standardisé de la sortie console



2.2.4.2. Dépendances

Ce composant représente les dépendances principales que nous utiliserons pour le projet



2.2.5. Couche d'accès aux données (Data Access Layer)

Cette couche sera implémentée au automatique à l'aide d'une implémentation JPA. JPA est l'interface Standard de java implémenter une solution de persistance de type ORM.

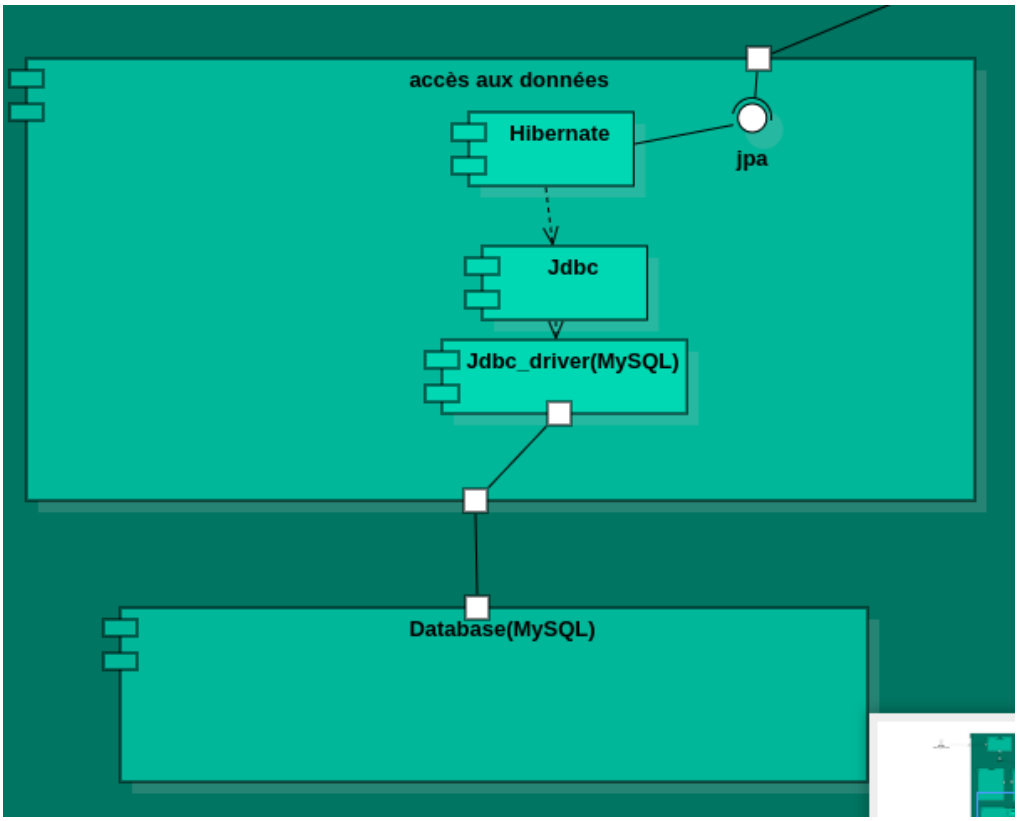
L'outil permet de mapper les objets que nous manipulons dans le paquetage entities dans un monde relationnel, en l'occurence ici notre base de données dans laquelle nous allons charger toutes les données de OpenFoodFacts ;

2.2.5.1. Choix technologiques

- Implémentation de l'interface JPA : Hibernate

  Jpa va dialoguer avec JDBC (Java Database Connectivity) pour atteindre la base de données.

- Système de Gestion de Base de Données Relationnelle : MySQL



3. Récapitulatif des solutions technologiques retenues pour le développement

- Langage : JAVA 17
- Moteur de production : Maven
- Interfaçage IHM : Console + GUI (Swing)
- Persistance des données : JPA – Hibernate
- Type de base de données : SGBDR
- Driver : MySQL
- Tests unitaires : Junit
- Logging : SLF4J (log-back)
- Versionning : Git, Github
- Progress Bar : ProgressBar, tong.me
- activation du cache : ehcache

4. Annexes