



Documentation Technique

WhoFeedsMe

Constantin Verine – 05.2019 – V1.0

1 TABLE DES MATIERES

2	Introduction	5
3	Cahier des charges	5
3.1	Objectifs	5
3.2	Spécifications	5
3.3	Environnement.....	5
3.4	Organisation.....	6
3.5	Livrables	6
3.6	Méthodologie	6
4	Analyse fonctionnelle.....	7
4.1	Prérequis	7
4.2	Use case	7
4.3	Fonctionnalités.....	8
4.3.1	Application C#	8
4.3.2	Base de donnée MySql.....	9
4.4	Interface	10
5	Analyse organique.....	11
5.1	Base de donnée	11
5.1.1	MySql.....	11
5.1.2	Mongodb	12
5.2	Application C#.....	13
5.2.1	frmWhoFeedsMeController	13
5.2.2	ModelWhoFeedsMe.....	13
5.2.3	MySqlDB.....	13
5.2.4	MongoDBClass	13
5.2.5	PRODUCT ET PRODUCTFACTORY	14
5.2.6	Informations de connexion	14
6	Tests	15
6.1	MysqlTests	15
6.1.1	IsConnectionOpened() et IsConnectionClosed().....	15
6.1.2	CRUD().....	15
6.2	MongoDBTests.....	15
6.2.1	ISCONNECTED().....	15

6.3	ModelWhoFeedsMeTests	15
6.3.1	IdentifyBrand() et IdentifyGroupBrand()	15
6.4	ProductTests	15
6.4.1	IsValid() et IsNotValid()	15
6.5	ProductFactoryTests	15
6.5.1	CreateProduct()	15
6.5.2	Normalize()	15
6.6	Test de performance	16
7	Planning	16
8	Conclusion	17
8.1	Problèmes rencontrés	17
8.1.1	Ressource humaine	17
8.1.2	Problèmes de version	17
8.1.3	Problemes d'execution	17
8.2	Améliorations	17
8.2.1	Barre de progression	17
8.2.2	Cd_Country	17
8.3	Expérience personnelle	17
9	Table des illustrations	18
10	Bibliographie	18
11	Annexes	19
11.1	Planning	19
11.1.1	Prévisionnel	19
11.1.2	Effectif	20
11.2	MySql	21
11.2.1	Base de données	21
11.2.2	Insertion de données	21
11.3	Code C#	21
11.3.1	Controller	21
11.3.2	Model	21
11.3.3	Classe MongoDB	21
11.3.4	Classe MySQL	21
11.3.5	Classes Product et ProductFactory	21

11.3.6 Tests 21

2 INTRODUCTION

Cette documentation sert à expliquer et détailler le travail que j'ai effectué dans le cadre du TPI (Travail Pratique Individuel), il m'a été demandé de réaliser le projet « WhoFeedsMe » pour l'obtention de mon CFC.

Ce projet a pour but d'analyser et de classer des produits alimentaires en fonction de la marque et du groupe.

3 CAHIER DES CHARGES

3.1 OBJECTIFS

Pendant les 11 jours consacrés au TPI, mon objectif est de créer un ETL (Extract, Transform, Load) permettant de classer des produits en fonction de leur marque et de leur groupe.

Pour ce faire il me faudra analyser les données et les filtrer en fonctions de règles mises à ma disposition.

3.2 SPECIFICATIONS

La base dont on extrait les données est une base MongoDB ayant plus de 800 000 produits, cette base est celle d'OpenFoodFacts.

L'application de transformation des données doit être développée en C#.

La base dans laquelle on charge les données est une base MySQL qui doit au préalable être chargée avec les marques par groupes alimentaires.

3.3 ENVIRONNEMENT

- PC : Windows 10
- Base de donnée : MongoDB V4.0.9, MySQL (EasyPHP V14.1 puis V17.1 (pour PHPMyAdmin))
- Outil MySQL : MySQL Workbench
- IDE : Visual Studio (2015 au début, 2019 après)
- Navigateur web : Google Chrome
- Outils bureautiques divers : MSOffice, Visual Studio Code
- Profiler : JetBrains DotTrace
- Google Drive (pour la gestion des versions)

3.4 ORGANISATION

Élève :

- Constantin Verine, constantin.vrn@eduge.ch

Maitre d'apprentissage :

- Michaël Mathieu, edu-mathieum@eduge.ch

Experts :

- Philippe Fontaine, ph_fontaine@bluewin.ch
- Fredy Aegerter, fredy@aegerter.me

3.5 LIVRABLES

Pour la fin du TPI, le 23 mai 2019

- Planning Prévisionnel et effectif
- Rapport du projet
- Manuel utilisateur inclus dans le rapport de projet
- Journal de bord

3.6 METHODOLOGIE

La méthodologie utilisée pour la réalisation du mandat est celle dite « EXCEL monodéveloppeur »
Je me contente de suivre un planning réalisé sous EXCEL.

4 ANALYSE FONCTIONNELLE

4.1 PREREQUIS

Tout d'abord il m'a fallu me munir d'un disque SSD pour des raisons de performance (les disques mis à disposition n'étaient pas du tout assez performants).

Marche à suivre¹ :

Chargement Initial

- Télécharger MongoDB (community version)
- Créer variable system MONGODB et mettre dans le chemin le répertoire bin de mongod
- Télécharger la base de données OpenFoodFacts (le dump mongod) et l'extraire dans le répertoire mongod (prend du temps)
- Dans le répertoire mongod créer un dossier vide
- Dans le bin lancer « mongod --dbpath ../db » avec db le nom du dossier
- Aussi dans le bin faire « mongorestore -d off -c products ../dump/off/products.bson » (prend du temps)
- Télécharger les connecteurs MongoDB – C# et C# – MySQL
- Ajouter ceux-ci dans les références du projet C#

Démarrage de la base mongo

- Dans le bin lancer « mongod --dbpath ../db » avec db le nom du dossier

Optionnel (mais conseillé)

Télécharger Compass, c'est une interface graphique utilisateur permettant de visualiser les données et plus encore. Très utile dans le cas présent pour analyser certains produits OFF.

4.2 USE CASE

- CRUD
 - o Créer
 - o Lire
 - o Modifier
 - o Supprimer
- ETL
 - o Presser le bouton (tout le travail s'effectue en arrière-plan)

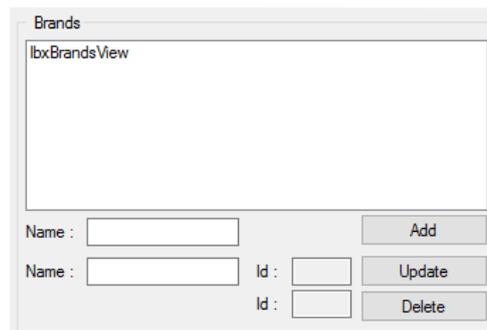
¹ Tous les liens sont trouvables dans la bibliographie

4.3 FONCTIONNALITES

4.3.1 APPLICATION C#

4.3.1.1 CRUD (CREATE READ UPDATE DELETE)

L'application C# permet de créer, lire, modifier et supprimer les données des groupes, marques et affiliations.

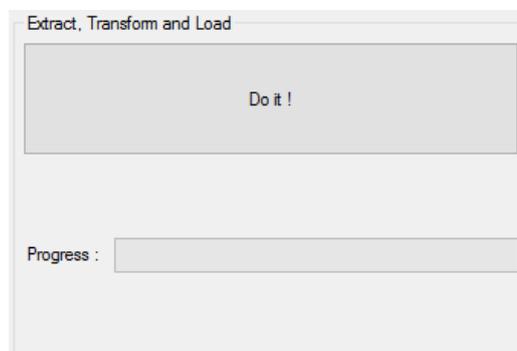


The screenshot shows a window titled "Brands". Inside the window, there is a large empty rectangular area labeled "IbxBrandsView". Below this area, there are two rows of input fields and buttons. The first row has a "Name" label followed by an empty text box and an "Add" button. The second row has a "Name" label followed by an empty text box, an "Id" label followed by an empty text box, and an "Update" button. Below the second row, there is another "Id" label followed by an empty text box and a "Delete" button.

Figure 1 CRUD

4.3.1.2 ETL (EXTRACT TRANSFORM LOAD)

La fonction principale de l'application est de charger une grande quantité de données, les traiter et les classer.



The screenshot shows a window titled "Extract, Transform and Load". Inside the window, there is a large rectangular area with a grey background and the text "Do it !" centered in it. Below this area, there is a "Progress :" label followed by an empty progress bar.

Figure 2 ETL

4.3.1.2.1 EXTRACT

L'extraction des données sous format Bson se fait de manière asynchrone pour ne pas surcharger la mémoire.

4.3.1.2.2 TRANSFORM

Tout d'abord on vérifie la validité des produits avec `IsValid()`, ceux qui nous intéressent sont ceux ayant un code bar EAN-13 ou EAN-8, les autres formats ne sont pas traités et les produits n'ayant pas de code sont également ignorés.

Certains produits apparaissent 2 fois dans la base, ils sont également ignorés lors de l'exécution du script.

Pour ces produits, on vérifie la marque, certains en ont plusieurs (en principe c'est la même marque mais pour un autre pays ou/et le groupe), on prend donc le premier libellé et on vérifie dans les associations à quel groupe appartient cette marque.

4.3.1.2.3 LOAD

Envoie des données vers la base de donnée MySQL.

4.3.1.3 NORMALIZE()

Cette fonction sert à remplacer toutes les lettres accentuées par des lettres normales, elle supprime aussi tous les caractères spéciaux ainsi que les espaces.

```
[1, Activa Capital]
[2, AXA]
[4, Belvédère]
[5, Bolton Group]
[6, Bonduelle]
[7, Bongrain]
[8, Campbell Soup Company]
[9, CapVest]
[11, CDC Capital]
```

Figure 4 Données normales



```
[1, ACTIVACAPITAL]
[2, AXA]
[4, BELVEDERE]
[5, BOLTONGROUP]
[6, BONDUELLE]
[7, BONGRAIN]
[8, CAMPBELLSOUPCOMPANY]
[9, CAPVEST]
[11, CDCCAPITAL]
```

Figure 3 Données normalisées

Lors de la transformation cette méthode est appelée 2 fois par produit et rend le programme beaucoup plus lent.

4.3.1.4 IDENTIFYBRAND() ET IDENTIFYGROUPBYBRAND()

`IdentifyBrand()` renvoie l'id de la marque du produit en paramètre. `IdentifyGroupByBrand()` par contre renvoie l'id du groupe en fonction de la marque.

4.3.1.5 AUTRE

Beaucoup d'autres fonctions mineures ont été implémentées mais pour la plupart elles ne servent qu'à appeler les méthodes situées plus haut ou bien sont utilitaires.

4.3.2 BASE DE DONNÉE MYSQL

La base de données MySQL sert à relier les marques aux groupes

4.3.2.1 INSERTION DE DONNÉES

Pour insérer les données demandées il m'a fallu aller sur la page Wikipédia donnée et faire une série de manipulations.

Tout d'abord je n'ai pris que les données appartenant au secteur de l'alimentation mais comme ce n'étais pas du tout suffisant il m'a fallu trouver un moyen de faire cela proprement et rapidement. Alors j'ai pensé à faire une application qui extrairait les données automatiquement mais je me suis rendu compte que cela ne valait pas la peine. J'ai donc extrait efficacement les données en utilisant les outils : Excel et Notepad++.

4.4 INTERFACE

Le premier jour de TPI j'ai imaginé une interface simple et efficace comme celle-ci-dessous et m'y suis conformé mais elle à subit quelques petites modifications durant la conception de l'application.

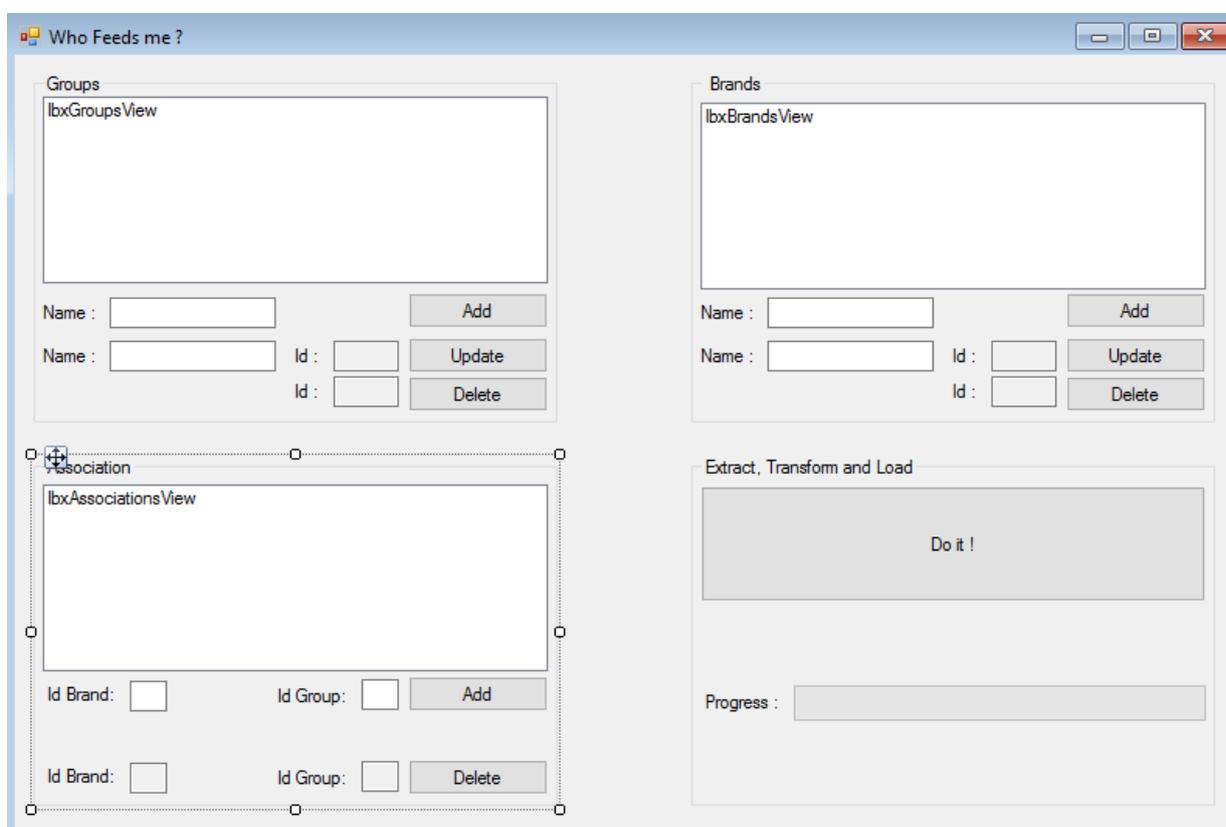


Figure 5 Forme principale

5 ANALYSE ORGANIQUE

5.1 BASE DE DONNEE

5.1.1 MYSQL

Le diagramme ci-dessous m'a été fourni par mon maître d'apprentissage, il a été implémenté avec succès, mais a subi quelques modifications en cours de route.

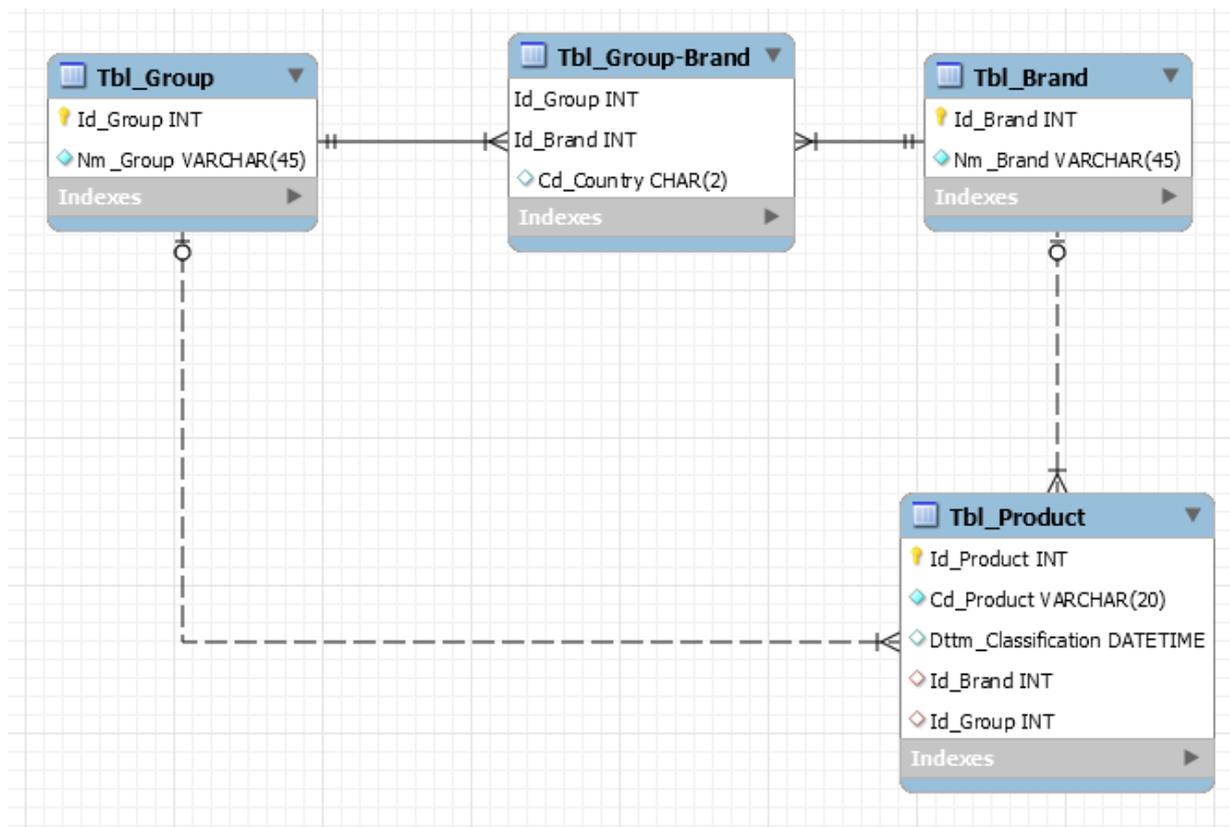


Figure 6 Base de données MySQL

5.1.1.1 TBL_GROUP ET TBL_BRAND

Ces tables servent uniquement à stocker le nom des groupes et des marques ainsi qu'à leurs attribuer un identifiant.

Table Name: Tbl_Group

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
Id_Group	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Nm_Group	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 7 Tbl_Group

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
Id_Brand	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Nm_Brand	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 8 Tbl_Brand

5.1.1.2 TBL_GROUP-BRAND

Cette table sert à associer les marques aux groupes et permet aussi de définir dans quel pays la marque a été déposée.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
Id_Group	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
Id_Brand	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
Cd_Country	CHAR(2)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 9 Tbl_Group-Brand

5.1.1.3 TBL_PRODUCT

Cette dernière table stock les produits et les affine à une marque et un groupe, la date et heure de l'affiliation sont aussi stockées.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
Id_Product	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Cd_Product	VARCHAR(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Dttm_Classification	DATETIME	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Id_Brand	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Id_Group	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 10 Tbl_Product

5.1.2 MONGODB

Les informations sur les données OFF se trouvent dans la bibliographie mais les deux champs exploités sont ceux-ci : code et brands_tags.

Ex: code: « 0009542028824 », brands_tags: [« Lindt », « Lindt & Sprungli (Schweiz) Ag »]

5.2 APPLICATION C#

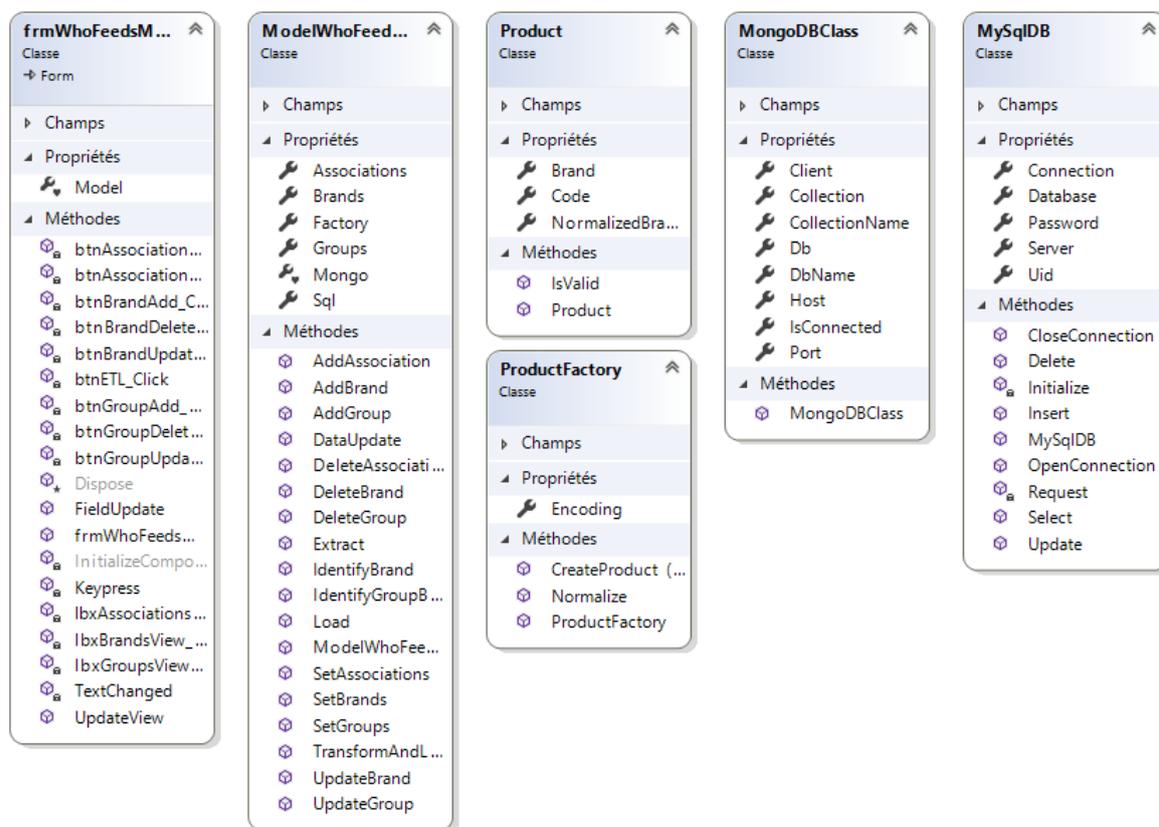


Figure 11 Diagramme de classes

5.2.1 FRMWHOFEEDESMECONTROLLER

Cette classe est celle du controller, elle crée une instance du modèle qu'elle charge et affiche, aussi elle traite tous les évènements comme les clics ou les sélections de données pour que l'application soit plus simple à utiliser.

5.2.2 MODELWHOFEEDESME

Celle-ci est le modèle de l'application, elle charge les données MySQL et permet tout types d'interactions avec ces dernières. Mais encore elle permet d'extraire les données mongoDB, les traiter et les envoyer vers MySQL.

5.2.3 MYSQLDB

Celle-là établie la connexion à la base MySQL et traite les entrées/sorties.

5.2.4 MONGODBCLASS

Cette-autre établie la connexion avec la base MongoDB

5.2.5 PRODUCT ET PRODUCTFACTORY

Ces deux classes transforment des données dites brutes en objet pour une utilisation simplifiée.

5.2.6 INFORMATIONS DE CONNEXION

Les Informations de connexion sont stockées dans un fichier App.config.

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <configuration>
3   <startup>
4     <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
5   </startup>
6   <appSettings>
7     <add key="server" value="localhost"/>
8     <add key="database" value="whofeedsme"/>
9     <add key="uid" value="admin_whofeedsme"/>
10    <add key="password" value="Super"/>
11    <add key="host" value="localhost"/>
12    <add key="port" value="27017"/>
13    <add key="dbName" value="off"/>
14    <add key="collection" value="products"/>
15  </appSettings>
16 </configuration>
```

Figure 12 Informations de connexion

6 TESTS

Les tests suivant ont tous été effectués grâce au projet de tests unitaires de Visual Studio.

6.1 MYSQLTESTS

6.1.1 ISCONNECTIONOPENED() ET ISCONNECTIONCLOSED()

Ces deux tests vérifient si la connexion à la base MySQL s'ouvre et se ferme convenablement.

6.1.2 CRUD()

Tout d'abord ce test crée une entrée puis regarde si elle a été créée. Ensuite il la modifie et regarde si elle a bien été modifiée. Finalement il supprime cette entrée.

6.2 MONGODBTESTS

6.2.1 ISCONNECTED()

Ce test vérifie si la connexion à la base mongo est établie.

6.3 MODELWHOFEEEDSMETESTS

6.3.1 IDENTIFYBRAND() ET IDENTIFYGROUPBRAND()

Ces tests vérifient si la bonne id est renvoyée.

Entrée : « kiri » Sortie : marque = 14, groupe = 3 soit groupe bel

6.4 PRODUCTTESTS

6.4.1 ISVALID() ET ISNOTVALID()

Deux tests pour la méthode IsValid() avec respectivement des valeurs valides et erronées.

6.5 PRODUCTFACTORYTESTS

6.5.1 CREATEPRODUCT()

Ce test sert à vérifier si le produit est bien créé et possède les bonnes valeurs.

6.5.2 NORMALIZE()

Ce test vérifie si la normalisation des infos fonctionne.

Test : " This îs à tèSt !!{}! " deviens : "THISISATEST".

6.6 TEST DE PERFORMANCE

Ce test fut effectué à la main plusieurs fois par jour mais n'était pas concluant car soit le programme prenait trop de temps à arriver au bout soit utilisait trop de ressources (le processeur montait à 15% rien qu'avec le programme).

Au final une exécution complète prend en moyenne 25 minutes et l'utilisation du processeur tourne autour des 5%.

7 PLANNING

Le planning prévisionnel ainsi que le planning effectif se trouvent en annexes.

On peut remarquer en comparant les deux plannings que outre les deux jours d'absences le planning a été suivi convenablement. On peut aussi voir que le temps attribué à certaines tâches était mal planifié.

8 CONCLUSION

8.1 PROBLEMES RENCONTRES

8.1.1 RESSOURCE HUMAINE

Le principal problème rencontré fut la maladie. Je suis effectivement tombé malade pendant ce travail et ai eu 2 jours de moins pour travailler sur le projet.

8.1.2 PROBLEMES DE VERSION

Pendant la conception du projet j'ai remarqué que j'étais limité par la version de deux logiciels fondamentaux : Visual Studio 2015 a été mis à jour en Visual Studio 2019 et EasyPHP-DevServer 14.1 a été mis à jour en EasyPHP-DevServer 17.1.

8.1.3 PROBLEMES D'EXECUTION

La fonction Normalize() prenait beaucoup de temps lorsqu'appelée plusieurs fois par produit, ce problème a été réglé en normalisant le groupe à la création d'un produit. La durée d'exécution du script a grandement diminué.

Aussi une mauvaise utilisations des dictionnaires a été corrigée, les clefs et valeurs ont été inversées ce qui a nettement amélioré les performances.

8.2 AMELIORATIONS

8.2.1 BARRE DE PROGRESSION

Lors de l'exécution de l'ETL on ne sait pas où en est le programme et vu que cela dure plusieurs minutes il serait intéressant d'ajouter une barre de progression.

8.2.2 CD_COUNTRY

Ce qui aurait dû être fait mais n'a pas pu être effectué :

Certaines marques portent un nom différent en fonction du pays ou bien appartiennent à un autre groupe. Il faudrait donc utiliser le champ countries_tags puis par exemple avec du regex prendre le/les identifiants ex : fr, en, ca, ru...

8.3 EXPERIENCE PERSONNELLE

Ce travail était très intéressant et m'a beaucoup apporté, j'ai vraiment apprécié la sensation de produire quelque chose d'utile, ce qui contraste bien avec le reste de ma formation.

Je suis vraiment déçu d'être tombé malade en plein milieu, sans cela j'aurais pu ajouter la classification par pays et serais encore plus satisfait du résultat.

9 TABLE DES ILLUSTRATIONS

Figure 1 CRUD.....	8
Figure 2 ETL.....	8
Figure 3 Données normalisées.....	9
Figure 4 Données normales.....	9
Figure 5 Forme principale.....	10
Figure 6 Base de données MySQL.....	11
Figure 7 Tbl_Group.....	11
Figure 8 Tbl_Brand.....	12
Figure 9 Tbl_Group-Brand	12
Figure 10 Tbl_Product.....	12
Figure 11 Diagramme de classes	13
Figure 12 Informations de connexion	14

10 BIBLIOGRAPHIE

MongoDB : <https://www.mongodb.com/download-center/community>

Base de donnée OpenFoodFacts : <https://world.openfoodfacts.org/data>

Connecteur C# MongoDB : http://mongodb.github.io/mongo-csharp-driver/2.7/getting_started/installation/

Compass pour MongoDB : <https://www.mongodb.com/download-center/compass>

Liste des marques par groupes alimentaires :

https://fr.wikipedia.org/wiki/Projet:Entreprises/Cartographie_des_marques_par_groupe

Infos sur les champs (OFF) : <https://world.openfoodfacts.org/data/data-fields.txt>

C# driver Références : <https://mongodb.github.io/mongo-csharp-driver/>

Forum pour developper : <https://stackoverflow.com/>

11 ANNEXES

Outre les plannings, les annexes ont leurs propres PDF.

11.1 PLANNING

11.1.1 PREVISIONNEL

Tâches à réaliser	Temps nécessaires	Jour 1	Jour 2	Jour 3	Jour 4	Jour 5	Jour 6	Jour 7	Jour 8	Jour 9	Jour 10	Jour 11	Total
Analyse et préparation													
Lecture et analyse de l'énoncé	00:15	00:15											00:15
Mise en place (impression & mise en page journal de bord)	00:15	00:15											00:15
Définition des tâches et remplissage du planning prévisionnel	01:30	01:30											01:30
Vérification de la configuration	01:00	01:00											01:00
Implémentation													
Création de la base MySQL	00:30	00:30											00:30
Création du script pour populer la base MySQL avec Données wikipédia	02:00	02:00											02:00
Création projet WhoFeedsMe avec les classes et les infos de connectio	00:20	00:20											00:20
Téléchargement connecteurs MySql et MongoDB	00:20	00:20											00:20
Conception Controller	00:20		00:20										00:20
Classe MySQLDB													
-Connexion avec la base mysql et déconnexion	01:30		01:30										01:30
-Select() simple	01:00		01:00										01:00
-Insert() simple	01:00		01:00										01:00
Classe MongoDB													
-Connexion à mongodb et verification	01:00		01:00										01:00
-Extract des données mongoDB en asynchrone (en dur)	02:00		01:10	00:50									02:00
Classe ModelWhoFeedsMe													
-Interaction avec les deux autres classes	00:30			00:30									00:30
-Envoie des données Mongoddb vers Mysql et vérification	01:00			01:00									01:00
Mise en place des filtres pour les données	10:00			03:00	03:00	03:00	01:00						10:00
Amélioration de Select() Insert() pour correspondre aux demandes	10:00				03:30	02:00	03:00	01:30					10:00
Amélioration Controller pour correspondre aux critères	04:00					01:00	00:30	00:30	02:00				04:00
load des données MongoDB vers MySQL (prend du temps à chaque fois	04:00			00:40	00:30	00:30	00:30	00:30	00:30	00:30	00:20		04:00
code des différentes methodes et algo	10:00						01:00	02:30	03:00	03:30			10:00
Tests	10:00							01:00		01:30	05:00	02:30	10:00
Documentation	20:00	01:20	01:30	01:30	00:30	01:00	01:30	01:30	02:00	02:00	02:10	05:00	20:00
Backup, vérification et planning effectif	05:30	00:30	00:30	00:30	00:30	00:30	00:30	00:30	00:30	00:30	00:30	00:30	05:30
Total	88:00:00	08:00	88:00:00										

11.1.2 EFFECTIF

Tâches à réaliser	Temps nécessaire	Jour 1	Jour 2	Jour 3	Jour 4	Jour 5	Jour 6	Jour 7	Jour 8	Jour 9	Jour 10	Jour 11	Total
Analyse et préparation													
Lecture et analyse de l'énoncé	00:15	00:15											00:15
Mise en place (impression & mise en page journal de bord)	00:15	00:15											00:15
Définition des tâches et remplissage du planning prévisionnel	01:30	01:30											01:30
Vérification de la configuration	01:00	01:00											01:00
Implémentation													
Création de la base MySQL	00:30	00:15											00:15
Création du script pour peupler la base MySQL avec Données wikipédia	02:00	01:15		03:00									04:15
Création projet WhoFeedsMe avec les classes et les infos de connectio	00:20	00:30											00:30
Téléchargement connecteurs MySql et MongoDB	00:20	00:30											00:30
Conception Controller	00:20		00:45										00:45
Classe MySQLDB													
-Connection avec la base mysql et déconnection	01:30		00:30										00:30
-Select() simple	01:00		01:30										01:30
-Insert() simple	01:00		00:10										00:10
Classe MongoDB													
-Connection à mongodb et verification	01:00												00:00
-Extract des données mongoDB en asynchrone (en dur)	02:00												00:00
Classe ModelWhoFeedsMe													
-Interaction avec les deux autres classes	00:30				00:30								00:30
-Envoie des données Mongoddb vers Mysql et vérification	01:00				03:10								03:10
Mise en place des filtres pour les données	10:00				03:30	03:00			01:00	00:45			08:15
Amélioration de CRUD pour correspondre aux demandes	10:00		01:00	00:40									01:40
Amélioration Controller pour correspondre aux critères	04:00		00:15	00:40	00:15	00:20							01:30
load des données MongoDB vers MySQL (prend du temps à chaque fois	04:00				00:30	00:20			00:40	00:30			02:00
code des différentes methodes et algo	10:00		02:40	02:10		02:10			03:00	02:00	02:00	01:30	15:30
Tests													
Documentation	20:00	02:00	01:00	01:20		02:00			01:35	02:30	02:00	01:00	07:05
Backup, vérification et planning effectif	05:30	00:30	00:10	00:10	00:05	00:10			00:15	00:15	00:05	01:30	03:10
Total	88:00:00	08:00	08:00	08:00	08:00	08:00	00:00	00:00	08:00	08:00	08:00	08:00	72:00:00

11.2 MYSQL

11.2.1 BASE DE DONNEES

2_whofeedsme.sql.pdf

11.2.2 INSERTION DE DONNEES

3_dataInsertion.sql.pdf

11.3 CODE C#

11.3.1 CONTROLLER

4_frmWhoFeedsMeController.cs.pdf

11.3.2 MODEL

5_ModelWhoFeedsMe.cs.pdf

11.3.3 CLASSE MONGODB

6_MongoDBClass.cs.pdf

11.3.4 CLASSE MYSQL

7_MySqlDB.cs.pdf

11.3.5 CLASSES PRODUCT ET PRODUCTFACTORY

8_Product.cs.pdf

9_ProductFactory.cs.pdf

11.3.6 TESTS

10_WhoFeedsMeTests.cs.pdf