CAUDRON CEDRIC

DOSSIER PROJET

Réalisation d'un algorithme dit de rapprochement, qui a pour but de détérminer le bon véhicule dans une base de données à partir de différentes caractéristiques.

Dossier de fin d'année de la formation Concepteur Développeur d'Applications à Valarep Valenciennes.

Réalisé dans le cadre de mon alternance chez CGI Finance, Marcq En Baroeul.

ANNEE 2020 / 2021

CGI FINANCE
GROUPE SOCIETE GENERALE

Table des matières

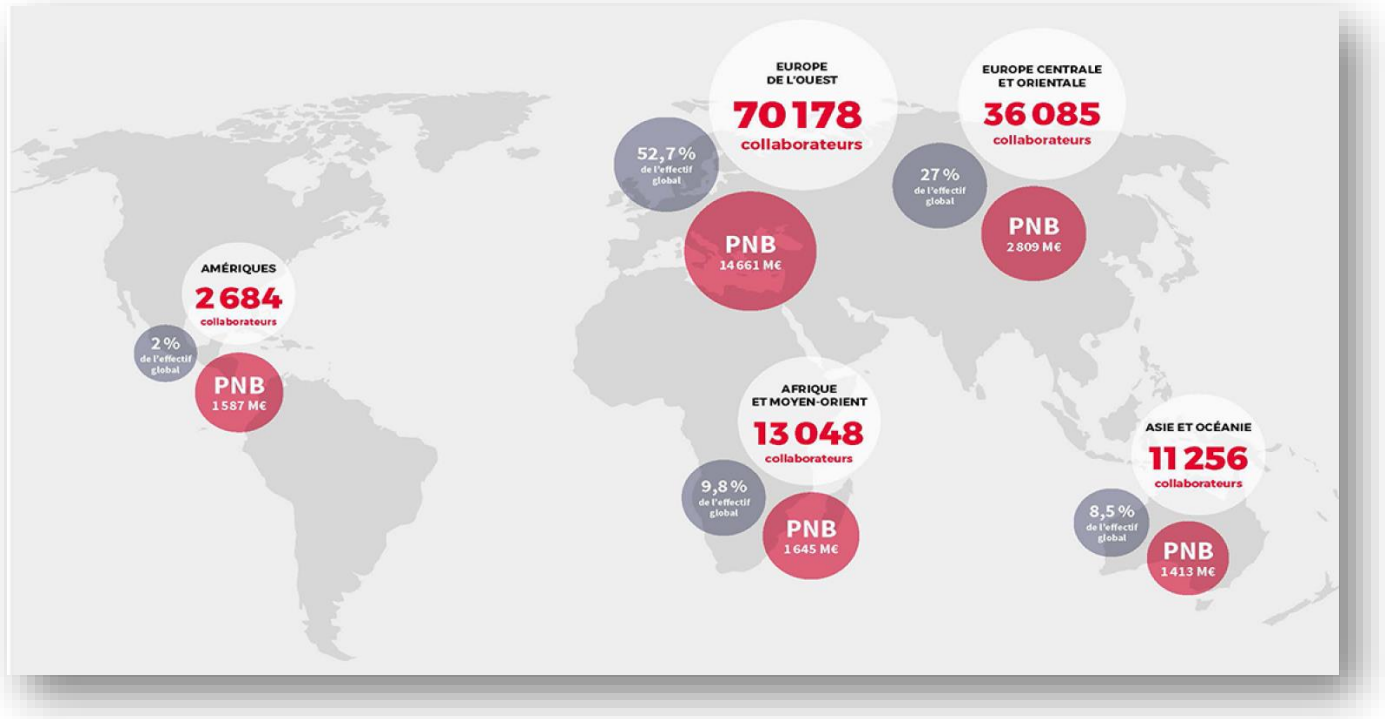
I : CGI FINANCE

1 : Présentation générale

1.1 : Le groupe Société Générale

La Société Générale est une des principales et une des plus ancienne banque française. En effet, le 4 mai 1864, Napoléon III signe le décret y donnant naissance, et elle est fondée par un groupe d'industriels et de financiers portés par des idéaux de progrès. Dès sa fondation, la banque nourrit l'ambition « de favoriser le développement du commerce et de l'industrie en France ».

Plus de 150 ans plus tard, elle est devenue un grand groupe financier international comportant plus de 30 millions de clients (particuliers, entreprises et investisseurs) et plus de 133 000 collaborateurs dans ses différentes filiales réparties dans 61 pays.



1.2 : CGI Finance

CGI Finance est un ensemble de sociétés faisant partie du Groupe Société Générale qui sont spécialisées dans le financement.

Elle compte plus de 1000 collaborateurs, 19 agences commerciales en France, et ce depuis plus de 70 ans.

Via son statut d'établissement de crédit, CGI Finance est composé :

- **D'une maison mère** : CGL (Compagnie Générale de Location d'Equipements) basée à Marcq en Baroeul
- **Et de multiples filiales** : Prioris, SGB Finance, Sefia et Concilian

Ses activités s'articulent autour de 3 axes :

- **Financement automobile** : (N°1 des financeurs indépendants en France)
- **Financement nautique** : (N°1 du financement nautique en Europe)
- **Regroupement de crédits**

2 : La DSQUAD

2.1 : Les services de développement

Au sein de CGI Finance, il existe deux services de développement logiciel. Le premier, la DOI, est le grand service qui développe et maintient les nombreuses applications mises à disposition des partenaires pour les aider dans leur activité, ainsi que les solutions internes.

De façon indépendante, il existe le service que j'ai intégré en octobre 2020, appelé la Dsquad. Il s'agit d'un petit service orienté plus spécifiquement sur les innovations.

Enfin, le tout est chapeauté par le service de sécurité informatique de la Société Générale. En effet, au vu des informations sensibles qui peuvent être échangées de par le statut de société financière, la sécurité y est draconienne afin d'éviter toute attaque, fuite de données etc.

2.2 : La Dsquad

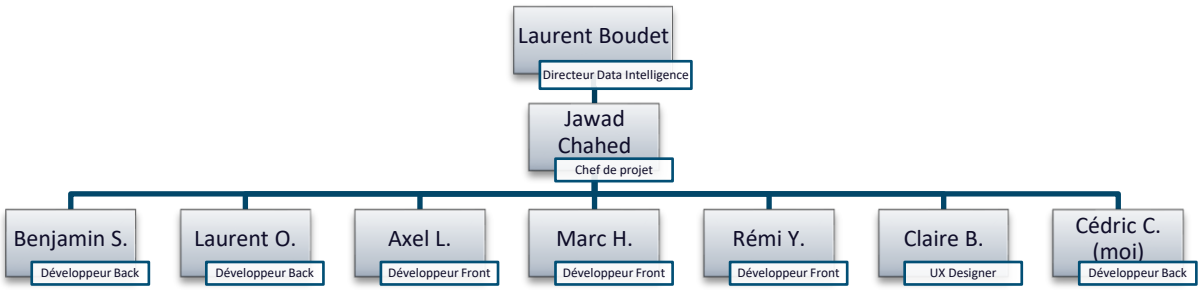
La Dsquad est donc un service informatique indépendant, comprenant peu de personnes (±10 personnes), créé à l'initiative de Laurent Boudet (Directeur Data Intelligence) et qui a pour but d'innover en créant de nouveaux outils qui pourront être partagés ensuite à l'ensemble de la société.

Depuis 2 ans, l'équipe développe une application mobile Android, nommée Occazio, de vente de véhicules entre particuliers, mais aussi via les concessionnaires partenaires. L'application met en œuvre nombre d'innovations pour améliorer au maximum l'expérience utilisateur, ainsi que la sécurité des vendeurs et acheteurs.

2.3 : Organigramme

Organigramme représentant les membres permanents de l'équipe. D'autres collaborateurs (de la DOI notamment) sont amenés à intervenir sur les différentes problématiques suivant les besoin.

Certains autres collaborateurs partagent leur temps entre la Dsquad et la DOI.



II : LES BESOINS

1 : Un financement au plus proche de la réalité

Une des principales activités de CGI Finance est donc le financement de véhicules d'occasion, vendus par les nombreux professionnels partenaires de la société. Une des principales problématiques dans ce secteur est de déterminer avec précision le prix du véhicule en fonction des informations que les partenaires nous envoient, qui ne sont pas standardisées.

L'algorithme servira à trouver le prix du véhicule neuf au plus proche de la réalité. Auquel sera appliqué une décote suivant son âge/kilométrage, puis ensuite sur cette base la mensualité et la VR (Valeur Résiduelle ou de Reprise) seront calculés.

Il faut savoir que, mécaniquement, plus la mensualité sera petite, plus la VR sera grande.

De plus le temps de la location, CGI sera propriétaire du véhicule.

La difficulté ici, c'est que pour rentrer dans ses frais, si on a une VR trop importante (et donc finalement des véhicules moins amortis avec une mensualité plus faible), sera la revente à terme si le client final ne le reprend pas. Car si la revente se fait à un prix inférieur à la VR, la société perdra de l'argent dans le cas où c'est elle qui a repris le véhicule.

Donc par exemple, si par rapport aux caractéristiques du véhicule, on rapproche un mauvais véhicule qui a un prix bien plus important que sa valeur réelle, déjà le client pourrait ne pas le prendre dû à une mensualité trop importante, et la VR sera mécaniquement trop élevée par rapport à la réalité du véhicule. Et donc la revente à ce prix difficile voire impossible. Ce qui mènera à terme à une vente à perte.

Il est donc important de déterminer avec le plus de précision et de certitude possible le prix neuf du véhicule, et c'est ici que j'entre en scène avec l'algorithme de rapprochement.

Ce n'est que le principe général du problème, car il existe différents cas de figures (comme par exemple lorsque le concessionnaire s'engage à le reprendre à tel prix, là ce sera à sa charge de le revendre. Et dans ce cas, CGI fera une préconisation de VR et de mensualité que le concessionnaire sera libre (dans une certaine mesure) de modifier).

2 : L'Argus, la base de référence

Il existe en France une base qui sert de référence dans l'industrie automobile pour recenser les différents véhicules existants, leurs prix, leurs caractéristiques et diverses informations utiles sur chaque véhicule commercialisé.

Or CGI Finance a fait l'acquisition de cette base de données sur les véhicules sortis les dix dernières années.

Ma principale tâche sera donc de développer un algorithme qui rapprochera les différentes caractéristiques des véhicules que nous fournissent les différents partenaires avec son identifiant dans la base Argus. Une fois obtenu, et même si la base Argus n'est pas exempte d'erreurs (je reviendrai plus tard sur la qualité générale des données), il sera très facile de récupérer le prix du véhicule neuf et ainsi de continuer le processus précédemment décrit de façon fiable.

3 : De nombreux partenaires, des données différentes

Comme dit précédemment, de nombreuses données nous parviennent de différents partenaires, de qualité inégale. Certains avec plus de données que d'autres, avec plus ou moins d'erreur. Ci-dessous quelques exemples (tronqués, j'ai enlevé plusieurs colonnes inutiles pour nous) des données qui devront à terme être rapprochées avec la base Argus.

Partenaire « Autosphère »

Immatriculation	CleArgusOID	T20_Partenaire_Code	T10_Concession_Code	PrixVenteTTC	TvaApplicable	DatePremiereMiseEnCirculation	Kilometrage	T10_Marque_Code	Marque_CodeCGI
ES401PE	1734127	AUTOSPHERE	futu	29900.00	1	2017-12-08 00:00:00.000	127479	BMW	BMW
FH646FP	2039085	AUTOSPHERE	fd08c2	20490.00	1	2019-06-25 00:00:00.000	23363	FORD	FOR
FJ812AV	2037524	AUTOSPHERE	fd95c6	19000.00	0	2019-07-23 00:00:00.000	8300	FORD	FOR
DZ063ZQ	1542714	AUTOSPHERE	geor	24999.00	0	2016-02-25 00:00:00.000	90434	PEUGEOT	PEU

Modele	T4_Energie_Code	Energie_CodeCGI	Co2	PuissanceFiscale	Couleur	phase	finition	portes	puissance_din	automatique	version
SERIE 5 TOURING	GSL	GSL	129	10	Mineralwess metallisee			5		1	520dA xDrive 190ch Sport Steptronic
FOCUS	ESS	ESS	126	7	BLANC METROPOLIS			5		1	1.0 EcoBoost 125ch BVA
FOCUS	ESS	ESS	107	6	METROPOLIS			5		0	1.0 EcoBoost 125ch
Traveller	GSL	GSL	139	8	Gris			4		0	2.0 BlueHDi 150ch Standard Active S&S

Partenaire "JeanLain"

Immatriculation	CleArgusOID	T20_Partenaire_Code	PrixVenteTTC	DatePremiereMiseEnCirculation	Kilometrage	T10_Marque_Code	Marque_CodeCGI	Modele	T4_Energie_Code	Energie_CodeCGI
FX532GX	2009557	VIVACAR	31299.00	2021-02-22 00:00:00.000	7000	audi	AUD	a1	esse	ESS
FX446GB	2370902	VIVACAR	48570.00	2021-02-19 00:00:00.000	6000	volkswagen	VOK	golf	dies	GSL
FX239FH		VIVACAR	24490.00	2021-02-18 00:00:00.000	6000	volkswagen	VOK	t cross	esse	ESS
FS239ZF	2164599	VIVACAR	33900.00	2020-09-28 00:00:00.000	6000	volkswagen	VOK	golf	elec	ELE

Co2	PuissanceFiscale	Couleur	phase	gamme	finition	portes	puissance_din	nb_rapports	automatique	version
137	6	rouge		A1 SPORTBACK	S line	5	116	7	1	A1 Sportback 30 TFSI 116 ch S tronic 7 S line
141	11	bleu		GOLF	GTD	5	200	7	1	Golf 2.0 TDI SCR 200 DSG7 GTD
134	6	bleu		T-CROSS	Active	5	110	6	0	T-Cross 1.0 TSI 110 Start/Stop BVM6 Active
46	8	blanc		GOLF	GTE	5	150	6	1	Golf Hybride Rechargeable 1.4 TSI 204 DSG6 GTE

Partenaire "ReezoCar"

powers_din	doors	emission_co2	year	weight	model	consumption	gearbox	body	brand	energy	title	powers_fiscal	country	mileage	price
181	5	109	2015	2260	v70	4,2	manual	estate	volvo	diesel			Nederland		12750
	5				rio		manual	saloon	kia	diesel	Kia Rio 1.5 CRDi 5p. UNICO PROPRIETARIO EURO 4		Italie	250000	
58	3		1999	795	twingo		manual	hatchback	renault	petrol	Renault Twingo - 1.2 Hélios apk elek.ramen elek.spiegels, centrale deurvergrendeling pano		Pays-Bas	165220	550
69	3	120	2019		500	5,2	manual	saloon	fiat	petrol	Fiat 500 III 2015 1.2 Pop 69cv my20	5	Italie	23486	9500

Partenaire "DCSNET"

date1mec	genre	marque	carrosserie	energie	puissancefiscal	nbplace	modele	kilometrage	prixventettc	prixintttc	gamme	version	nbporte	typeboitevitesse	co2	puissancedin	nbrapportboite	consomoy
28/05/2019	VP	RENAULT	BREAK	ES	6		MEGANE IV ESTATE BUSINESS MÃ©gane IV Estate TCe 115 FAP Business	6338	16490	16490	MEGANE IV ESTATE BUSINESS	MÃ©gane IV Estate TCe 115 FAP Business	5	0	124	115	6	5.3
25/10/2019	VP	PEUGEOT	TOUT-TERRAIN	ES	7		3008 Puretech 130ch S&S EAT8 GT Line	33410	28280	28280	3008	Puretech 130ch S&S EAT8 GT Line	5	1	112			0
25/09/2017	VP	VOLKSWAGEN	MONOSPACE	GO	8	5	TOURAN 2.0 TDI 150 BMT DSG6 5pl Carat	83345	21900	21900	TOURAN	2.0 TDI 150 BMT DSG6 5pl Carat	5	1	122	150	6	4.7
29/07/2016	VP	FIAT	CABRIOLET	ES	4	4	500C SERIE 4 500C 0.9 85 ch TwinAir S&S Lounge	68482	10490	10490	500C SERIE 4	500C 0.9 85 ch TwinAir S&S Lounge	2	0	90	85	5	3.8

Ce sont donc quelques exemples de données brutes que mon algorithme devra traiter afin de trouver le véhicule correspondant dans la base Argus. Notez que sur les deux premier il existe déjà un champ « CleArgusOID » qui donne l'information. Il s'agit de fichiers qui ont été retraités par la DOI et leur algorithme de rapprochement déjà en place, j'y reviendrai dans le chapitre suivant.

4 : Les solutions existantes

4.1 : Le rapprochement de la Dsquad

Actuellement, dans l'application Occazio, un algorithme est déjà en place faisant le rapprochement entre des caractéristiques de véhicules et leur Id Argus.

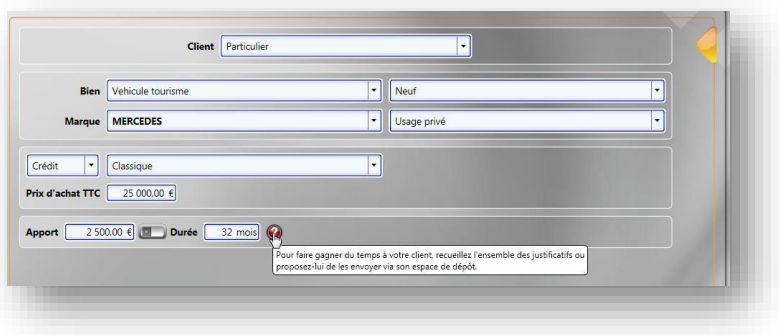
Développé en Python par mon prédécesseur, il est efficace sur des cas simples ayant une qualité de données en entrée suffisante. Ce qui pour l'usage de l'application est suffisant.

Néanmoins, pour des utilisations plus critiques comme la détermination d'une valeur résiduelle de véhicule, où l'on doit être sûr de ne pas se tromper au risque de faire perdre de l'argent à la société, il montre vite ses limites.

Il m'a été demandé dès mon premier jour dans l'équipe d'analyser ce code, sachant qu'il sera remplacé par la nouvelle version dès qu'elle sera assez performante.

L'algorithme se base essentiellement sur du nettoyage de données ainsi que le calcul d'une distance de Levenshtein qui permettra de classer les candidats avec une probabilité.

4.2 : Le rapprochement de la DOI



La DOI quant à elle a également sa solution pour le rapprochement. Son algorithme est intégré dans un projet bien plus vaste, Vivafi, développé en C#, qui est un outil de pricing et d'acceptation de financement mis à disposition des concessionnaires partenaires.

J'ai été amené également à analyser le code de cette solution. Qui présente des soucis dans ses résultats ainsi que dans sa maintenabilité. Le problème étant principalement dû au fait qu'il a été développé pour répondre au formatage des données d'un partenaire en particulier (et même il existe des erreurs dans ses résultats j'y reviendrai plus tard).

Dans les données brutes vues plus haut, l'id Argus renvoyé dans le tableau est celui rapproché par cet algorithme. Ce qui est fort utile pour la suite car cela permettra de comparer les résultats entre cette solution et la nouvelle une fois développée.

5 : Objectif

Ainsi, au vu de ces problématiques, mon objectif cette année est de :

- Concevoir et développer un algorithme qui rapprochera les caractéristiques de véhicules avec l'id Argus qui lui correspondra.
- De faire en sorte qu'il soit fiable afin d'éviter les problèmes lors de la cotation, donc du calcul des mensualités et de la Valeur Résiduelle pour les demandes de LOA et LDD.
- De le généraliser au maximum pour qu'il soit utilisé avec le maximum de partenaires toujours dans le but d'obtenir les résultats les plus fiables possibles.
- Déployer, avec le reste de l'équipe, la solution développée pour une utilisation dans l'application Occazio.
- Faire en sorte qu'il puisse être utilisé par les outils plus généraux de la société s'ils en ont besoin.

III : LES SOLUTIONS ALGORITHMIQUES

1 : Les dimensions

Un véhicule est un objet complexe qui a nombre de caractéristiques. Donc la première étape sera, afin de comparer ce qui est comparable, de découper chaque véhicule en une liste de ses différentes caractéristiques. Chaque caractéristique sera donc incluse dans un sous objet que nous nommerons « dimension ».

L'avantage de ce système sera que nous pourrons, à tout moment, comparer telle ou telle caractéristique d'un véhicule avec ceux issues de la base Argus (qui seront aussi découpés de la même façon), sans risque d'avoir des erreurs en essayant de comparer des caractéristiques qui n'ont rien à voir entre elles.

De plus, à la création de chaque dimension, nous pourrons déjà appliquer des premiers traitements de nettoyages de données, comme la suppression des caractères spéciaux, mise en majuscule, ainsi que l'application de transcodifications (je reviendrai plus en détail sur ce système dans la partie technique) sur certaines données.

Ainsi, un véhicule, à l'heure où j'écris ces lignes car c'est susceptible d'évoluer, sera décomposé en 22 dimensions :

- Dimension Marque
- Dimension Modèle
- Dimension Sous-Modèle
- Dimension Date MEC (Mise En Circulation)
- Dimension Energie
- Dimension Boite de Vitesse
- Dimension Type Boite de Vitesse
- Dimension Puissance DIN
- Dimension Puissance Fiscale
- Dimension Nombre de Places
- Dimension Nombre de Portes
- Dimension CO2
- Dimension Carrosserie
- Dimension Poids
- Dimension Libellé
- Dimension Finition
- Dimension Phase
- Dimension Génération
- Dimension Consommation
- Dimension Prix HT
- Dimension Prix TTC
- Dimension Cm3

2 : Les distances

2.1 : Principe général

Après avoir créé les dimensions pour le véhicule à tester, ainsi que celles des véhicules candidats issus du référentiel Argus, il faut pouvoir comparer chacune d'entre elles pour établir un classement.

Un système de distance a donc été choisi. Cela se présente comme une valeur numérique, comprise entre 0 et 1, qui sera affecté à chaque dimension du véhicule candidat. La valeur 0 signifiant une correspondance parfaite, et plus la valeur s'approchera de 1, plus les caractéristiques seront différentes.

De plus, pour chaque type de dimension nous affecteront une pondération qui nous servira plus tard pour le calcul de la distance générale. Cela permettra de donner plus de poids à une caractéristique jugée plus importante que les autres, ainsi que d'affiner les résultats suivant les différents partenaires si besoin est.

2.2 : Distance discrète

Cette première méthode de calcul de distance s'appliquera à chaque dimension. Elle consiste en la vérification de l'égalité parfaite entre la dimension à tester et celle du candidat.

Ainsi, si le résultat est positif, cette distance sera égale à 0. Sinon on va procéder aux autres traitements.

2.3 : Distances sur dimensions numériques

Dans les cas de dimensions numériques (comme le nombre de portes, de places, les puissances fiscales et DIN etc.…) , nous allons utiliser la distance euclidienne classique. Simple et efficace pour la plupart des cas.

Méthode de calcul :

$$\forall\, \boldsymbol{vRef} \Rightarrow \boldsymbol{distance} = \frac{|\boldsymbol{v} - \boldsymbol{vRef}|}{\boldsymbol{vMax}}$$

$\boldsymbol{v} = \boldsymbol{valeur}$ à $\boldsymbol{tester}$
$\boldsymbol{vRef} = \boldsymbol{valeur\ référentiel}$
$\boldsymbol{vMax} = \boldsymbol{valeur\ la\ plus\ grande\ entre\ v\ et\ vRef}$

2.4 : Distances sur dimensions textuelles

Pour les dimensions textuelles, la distance discrète sera principalement utilisée, après transcodification et nettoyage si besoin.

2.5 : Cas particulier, distance sur le libellé

Le libellé est une dimension textuelle complexe, comprenant nombre d'informations qui peuvent être gérées par d'autres dimensions. Ainsi pour lui un traitement bien plus complexe doit être appliqué.

La première étape le concernant sera d'extraire les informations qu'il contient afin de les mettre dans leurs dimensions respectives, et de les nettoyer du libellé lui-même.

Prenons un exemple de libellé de véhicule :

« Tiguan 2.0 TDI 150ch Carat Exclusive 4Motion »

Dans cet exemple, il me faudra donc extraire :

- « Tiguan » qui correspond au modèle
- « 150ch » qui correspond à la puissance DIN
- Et « Carat Exclusive » qui correspond à la finition

Pour toutes ces informations extraites, il me faudra les nettoyer du libellé peupler leurs dimensions respectives (si elles ne sont pas renseignées en entrée car l'on part du postulat que si une donnée est en entrée elle est potentiellement plus fiable) et ainsi suivre le processus normal de calcul de distances pour ces dimensions. Il faudra également nettoyer les libellés du référentiel dans le cas où ces informations y figurent également (exactement les mêmes) afin de calculer ensuite la distance sur le libellé.

Ceci a une valeur discriminante supplémentaire car si dans le référentiel les infos du libellé ne sont pas les mêmes, je ne les nettoie pas. Donc cela créera de plus fortes différences dans le cas de véhicules différents.

Une fois les libellés nettoyés, il y a deux calculs de distance sur cette dimension :

- **Distance « mot par mot » :**

$$\forall\, \boldsymbol{vRef} \Rightarrow \boldsymbol{distance} = \mathbf{1} - \frac{\boldsymbol{nbre\ mots\ du\ libRef\ compris\ dans\ libTest}}{\boldsymbol{nbre\ mot\ max}}$$

$$\boldsymbol{libRef} = \boldsymbol{libell\acute{e}\ du\ r\acute{e}f\acute{e}rentiel}$$
$$\boldsymbol{libTest} = \boldsymbol{libell\acute{e}\ du\ v\acute{e}hicule}\ \grave{a}\ \boldsymbol{tester}$$

Ceci permet d'obtenir un ratio entre le nombre de mot présent dans les deux libellés à comparer.

- **Distance de Levenshtein :**

J'y applique en sus une distance de Levenshtein. C'est un indicateur qui représente le nombre d'opérations qu'il est nécessaire d'effectuer pour transformer une chaine de caractères en une autre.

Pour ramener le résultat entre 0 et 1, je divise cette distance par le nombre de caractères du libellé le plus long entre les deux à comparer.

$$\forall\, \boldsymbol{vRef} \Rightarrow\ \boldsymbol{distance} = \frac{\boldsymbol{Lev(libRef, libTest)}}{\boldsymbol{nbre\ char\ max}}$$

$$\boldsymbol{libRef} = \boldsymbol{libell\acute{e}\ du\ r\acute{e}f\acute{e}rentiel}$$
$$\boldsymbol{libTest} = \boldsymbol{libell\acute{e}\ du\ v\acute{e}hicule}\ \grave{a}\ \boldsymbol{tester}$$

- **Calcul de la distance des libellés :**

Il a été établi d'appliquer des pondérations sur chacune de ces deux distances concernant le libellé. Ainsi , son calcul est la moyenne pondérée des distances « mot par mot » et de Levenshtein.

$$\forall\, \boldsymbol{vRef} \Rightarrow\ \boldsymbol{distance} = \frac{\boldsymbol{pLev} * \boldsymbol{dLev} + \boldsymbol{pMo} * \boldsymbol{dMo}}{\boldsymbol{pLev} + \boldsymbol{pMo}}$$

$$\boldsymbol{pLev} = \boldsymbol{pond\acute{e}ration\ Levenshtein\ du\ libell\acute{e}}$$
$$\boldsymbol{dLev} = \boldsymbol{distance\ de\ Levenshtein}$$
$$\boldsymbol{pMo} = \boldsymbol{pond\acute{e}ration}\ \text{"}\boldsymbol{Mot\ par\ Mot}\text{"}\ \boldsymbol{du\ libell\acute{e}}$$
$$\boldsymbol{dMo} = \boldsymbol{distance}\ \text{"}\boldsymbol{Mot\ par\ Mot}\text{"}$$

2.6 : Calcul de la distance générale

Enfin, pour terminer cette partie théorique, il nous reste à calculer la distance de chaque véhicule du référentiel avec celui à tester.

Pour se faire, nous calculons simplement pour chaque véhicule la somme pondérée de ses distances.

$$\forall\, \boldsymbol{vRef} \Rightarrow\ \boldsymbol{distance} = \boldsymbol{p1} * \boldsymbol{d1} + \boldsymbol{p2} * \boldsymbol{d2} + \cdots + \boldsymbol{pn} * \boldsymbol{dn}$$
$$\boldsymbol{p1} \ldots \boldsymbol{pn} = \boldsymbol{pond\acute{e}ration\ de\ chaque\ dimension}$$
$$\boldsymbol{d1} \ldots \boldsymbol{dn} = \boldsymbol{distance\ de\ chaque\ dimension}$$

Et pour finir, il suffit de classer les véhicules en fonction de leur distance générale croissante pour avoir en théorie un classement des véhicules, des plus proches de celui à tester aux plus éloignés.

IV : MISE EN OEUVRE

1 : L'environnement

1.1 : Architecture cible

Dans un premier temps l'algorithme de rapprochement devra être développé sous la forme d'une API en ASP.NET 3.1 avec deux end points parallèles. Un de type REST et l'autre de type gRPC, devant à terme être déployé sur un environnement Azure Cloud.

Au vu des dernières discussions au niveau de l'architecture, nous nous orienterions actuellement plutôt vers une mise à disposition sous la forme d'un package Nuget. Néanmoins, ayant été développé sous la première forme au jour d'aujourd'hui, je n'aborderai pas ce point dans ce dossier.

Azure est une offre d'hébergement et de services proposée par Microsoft.

Elle propose à l'heure actuelle plus de 200 produits afin de répondre aux besoins des entreprises (ou autre) dans la gestion de leurs données, déploiement d'applications etc.…

C# est un langage de programmation orienté objet, fortement typé et dérivé du C++, développé et commercialisé par Microsoft depuis 2002.

Il permet le développement sur la plateforme Microsoft .NET, et en particulier ici sur ASP.NET.

ASP.NET Core est un framework Web, gratuit et open-source, développé par Microsoft.

Il est une réécriture complète qui unifie ASP.NET MVC et ASP.NET API Web en un seul modèle de programmation. Il permet, entre autres, la génération et l'exécution multi-plateforme sur Windows, Mac et Linux.

Architecture REST, ou REpresentational State Transfert, est un standard créé en 2000 par Roy Fielding, informaticien américain. Son but est de faire transiter une représentation de l'état (les données) de notre application vers d'autres applications (ou inversement). En utilisant différents formats de données possible (JSON (ce qui sera notre cas), XML, CSV, …).

On accède aux ressources via des URLs uniques. Et l'on peut appliquer des modifications aux ressources, ajouter une nouvelle ressource, ou supprimer des ressources via les différents verbes HTTP. Dans notre cas ce point diffère car il s'agira de l'envoi de données (un véhicule) en POST afin d'obtenir un résultat après traitement.

Le standard REST répond à plusieurs critères :

- Client – Server : Un mode de communication avec séparation de rôles entre client et serveur
- Stateless Server : Les requêtes doivent contenir toutes les informations nécessaires au traitement. Le serveur ne connait donc pas l'état du client entre deux requêtes.
- Cache : La réponse du serveur doit être cacheable coté client.
- Uniform Interface : La méthode de communication entre client et serveur doit être uniforme avec des ressources identifiables. Ainsi, en http, la requête est identifiée avec une URL, et la réponse comprend un body et une entête.
- Layered System : Le système doit permettre le rajout de couches intermédiaires (proxy, firewall, CDN, etc.…).
- Code-on-Demand Architecture (optionnel) : L'architecture doit permettre d'exécuter du code coté client.

gRPC est un framework RPC (Remote Procedure Call) open source initialement développé par Google utilisant le protocole HTTP/2 pour le transport des données.

Il permet une communication sécurisée (avec son système d'identification intégré) et plus rapide entre le client et le serveur, et est utilisé dans notre cas pour les appels vers l'API par d'autres services.

En revanche, contrairement au standard REST, les données ne sont pas human-readable.

1.2: Les outils utilisés

Pour mener à bien le développement de l'algorithme, ainsi que l'analyse des résultats, j'ai été amené à utiliser nombre de logiciels et d'outils dont voici une liste (non-exhaustive).

Visual Studio 2019 professionnal :

IDE phare de l'environnement de développement de Microsoft. Outil principal qui m'aura permis de coder l'application.

Git :

Git est un logiciel de gestion de version. Il permet de stocker tout l'historique du code écrit sur un projet, de visualiser les modifications faites au cours du temps, de revenir en arrière en cas de problème. Il simplifie également le travail en équipe grâce au système de branche.

Sourcetree :

Logiciel gratuit permettant de gérer plus simplement le versionning Git grâce à une interface graphique.

PostgreSQL :

Système de Gestion de Base de Données Relationnelle. Utilisé car les solutions existantes dans la société l'utilisent pour gérer les données du référentiel Argus. Dans mon cas, il tournait dans un premier temps dans un container Docker, lui-même sur une distribution Ubuntu dans une VM sur mon poste de travail. Plus tard, à la faveur d'une mise à jour du référentiel, j'ai recréé la base en local sur mon poste, ce qui était bien moins lourd à gérer (même si plus éloigné de l'environnement de production). A terme, et c'est ce qui justifie ce choix, l'algorithme ne communiquera plus directement avec cette base mais avec une nouvelle API (en cours de développement) qui gérera spécifiquement les besoins d'accès aux données du référentiel Argus.

DBeaver :

Logiciel libre sous la licence Apache permettant l'administration d'un grand nombre de système de base de données différentes.

Postman :

Logiciel permettant d'effectuer des requêtes et de visualiser les réponses obtenues. Il m'aura surtout servi pour vérifier que la communication fonctionne vers l'Endpoint REST.

Excel :

Tableur faisant partie de la suite Microsoft Office. Il aura été primordial dans l'analyse des données. Que ce soit en entrée ou en sortie. En effet, les données en entrée m'étaient données au format csv. Et les résultats je les écrivais dans ce même format. Ce qui m'aura permis de créer des tableaux croisés dynamiques afin d'établir différentes statistiques. Sur ce projet l'analyse a été très importante et m'aura pris la majorité de mon temps.

Teams :

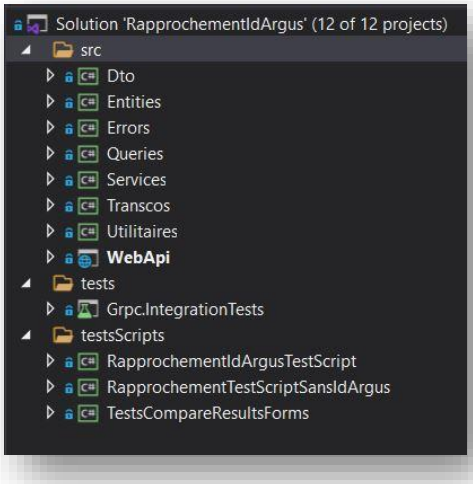
Le télétravail ayant été de mise pendant cette année, la communication se faisait essentiellement par Teams. Chaque Daily chaque matin, ainsi que les réunions plus spécifiques notamment pour la présentation et l'analyse de mes résultats.

Trello :

L'équipe ayant une organisation de type AGILE, les différentes tâches sont réparties aux différentes personnes de l'équipe à l'intérieur de plusieurs tableaux Trello. Concernant le rapprochement, j'en ai un que j'organise comme je le veux, étant seul sur le développement de cette API en particulier.

2 : Architecture du code

2.1: Architecture générale



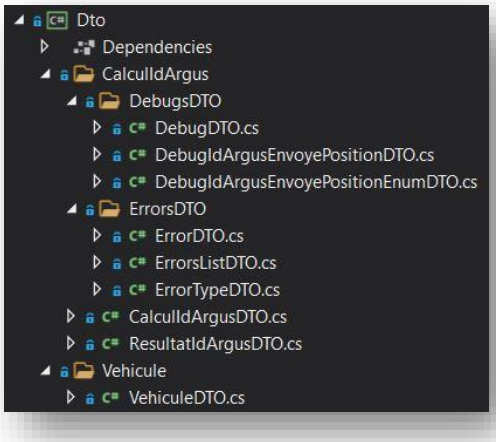
La solution a été divisée en 12 projets différents afin de séparer les responsabilités des différentes parties de l'algorithme.

Dans un dossier src, 8 projets qui, ensemble, composent l'algorithme en tant que tel sous sa forme d'API.

Dans un dossier tests, nous aurons tous les projets de tests. Actuellement il n'y en qu'un pour des tests d'intégration pour la communication gRPC.

Enfin, le dernier dossier contient différents scripts qui serviront de clients automatisés de l'API, récupérant les caractéristiques des véhicules dans différents fichiers, les envoyant à l'API, réceptionnant les réponses, les traitant et écrivant des fichiers .csv afin de pouvoir analyser et créer des statistiques sur les données récupérées.

2.2: Projet DTO

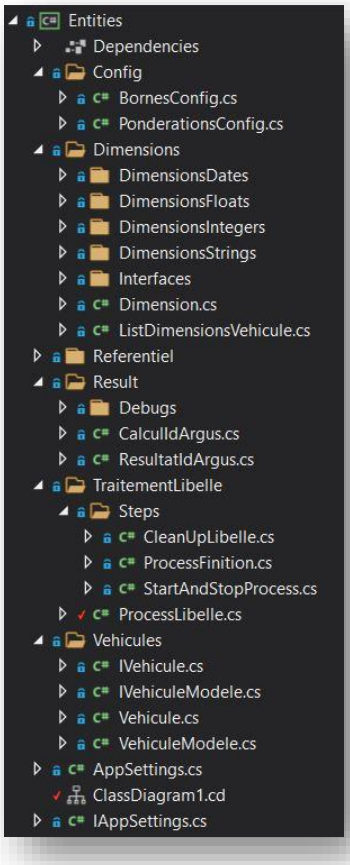


Ce projet contient toutes les classes qui permettront d'instancier les objets qui seront exposés par l'API. Ce sont les DTO (Data Transfer Object).

L'API donc, à chaque appel, ne recevra qu'un VehiculeDTO en entrée, et ne renverra qu'un CalculIdArgusDTO.

Aucune opération n'est effectuée dans ces objets dont leur seul but est le transport des données.

2.3: Projet Entities



Ce projet va regrouper toutes les classes nécessaires au fonctionnement interne de l'algorithme.

On y retrouve des « doubles » des classes présentes dans le DTO, et la transformation de l'une en l'autre se fera par mapping.

On y trouve également des classes de configuration dans lesquelles seront externalisés les pondérations et toutes constante dont j'aurais besoin. L'emplacement définitif de toutes ces configurations est encore à définir.

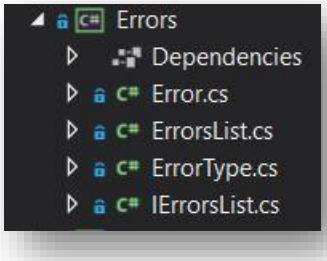
On y retrouve également toutes les classes liées aux différentes dimensions, regroupées par type.

Ensuite viennent les classes liées à Entity Framework (l'ORM le plus utilisé en ASP.NET) afin de communiquer avec la base référentielle de l'Argus.

Des classes servant au traitement spécifique du libellé.

Et enfin, une classe AppSettings et son interface qui feront le lien avec différentes configurations (comme les chaines de connexions).

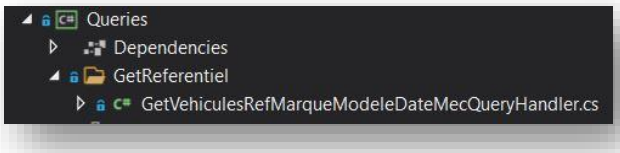
2.4: Projet Errors



Ce projet regroupe tout un système de gestion des erreurs « maison » pour l'ensemble de l'algorithme.

Il permettra, grâce à l'injection de dépendance, de conserver toutes les erreurs potentielles au cours du traitement des données, avec une gestion de leur différents types et importances.

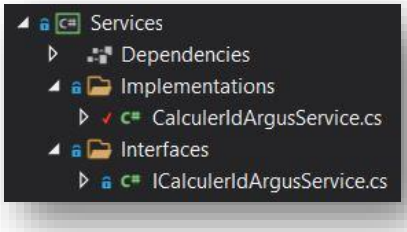
2.5: Projet Queries



Projet qui servira pour la communication avec la base de données du référentiel Argus.

C'est ici que l'on retrouvera les requêtes et certains traitements afin d'obtenir la liste des véhicules candidats en fonction de leur marque / modèle / date de mise en circulation.

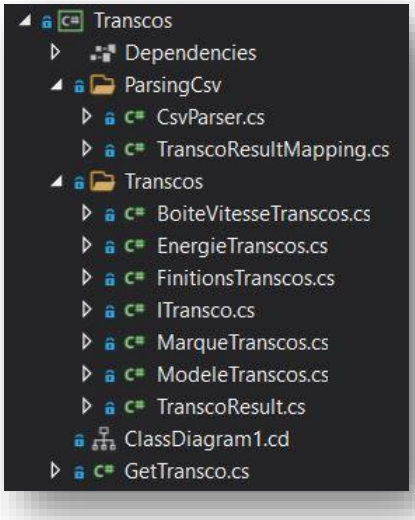
2.6: Projet Services



Ce projet est celui qui contiendra le service de l'algorithme.

Il s'agit du « chef d'orchestre », qui définira les opérations à effectuer, dans quel ordre, et renverra la réponse au controller à la fin du traitement.

2.7: Projet Transcos



Ce projet est celui qui gérera la lecture et la restitution des différents types de transcodifications dont nous pourrons avoir besoin dans le traitement.

Son principe est simple. Il permet, quand il est totalement impossible de retrouver de façon automatique une valeur, nous appliquons une transcodification sous la forme de correspondance clé-valeur. La clé pouvant être dupliquée pour correspondre à plusieurs valeurs et vice-versa. Il est fortement utile par exemple dans le cas des boites de vitesse qui sont souvent envoyées sous la forme textuelle, alors qu'elles sont représentées sous forme d'un code dans le référentiel Argus.

Les tables de transcodifications sont ici enregistrées sous forme de fichier .csv (imposé).

Le souci avec ce système est qu'il demande une maintenance humaine pour trouver et rajouter les clé-valeur manquantes si besoin.

```
GPL;GPL
GPL;GNV
GPL;Gaz
lpg;GPL
GNV;GNV
Gaz;Gaz
Gaz;GPL
Gaz;GNV
Diesel;Diesel
dies;Diesel
GO;Diesel
GSL;Diesel
gasoil;Diesel
```

```
BVRD;BVRD
BVRD5;BVRD5
BVRD6;BVRD6
BVRD7;BVRD7
Automatic;CVT
Automatique;CVT
Boite automat.;CVT
```

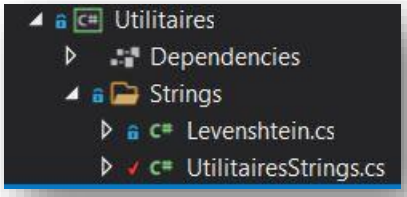
Exemples de tables de transcodifications avec, à gauche, un extrait de celle concernant l'énergie, et à droite, les boites de vitesses.

On peut y voir la duplication des clés ou valeurs. Ainsi, par exemple, si dans le véhicule envoyé nous avons une énergie « Gaz », les valeurs possibles seront « Gaz », « GPL » ou « GNV », qui correspondent à ce que l'on peut trouver pour les véhicules roulant au gaz dans le référentiel.

De même, si la boite envoyée est « Automatic », « Automatique » ou « Boite automat. », nous aurons comme valeur le code « CVT ».

Ce projet permet donc la récupération, par parsing du fichier csv, des différentes valeurs possibles correspondant à la valeur envoyée.

2.8: Projet Utilitaires

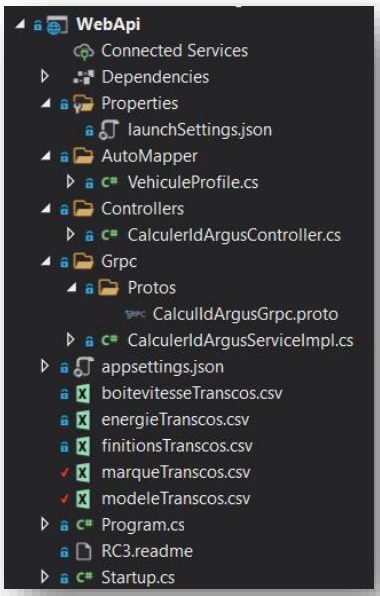


Petit projet comprenant un ensemble de classes statiques servant, pour la plupart, au nettoyage des chaines de caractères (enlever les espaces, les caractères spéciaux etc.…)

Dans celui-ci également qu'est calculé la distance de Levenshtein entre deux chaines de caractères.

C'est un petit projet que j'ai externalisé car il me sert dans différents travaux et pas seulement pour celui-ci.

2.9: Projet WebApi



Enfin, le dernier projet de la solution est la WebApi en tant que tel. Point d'entrée de l'application, c'est celui qui contiendra les controllers (type REST et gRPC), le système de mapping entre les objets (notamment pour transformer les DTO en Entités), toute la configuration générale de l'API comme le système permettant l'injection de dépendance.

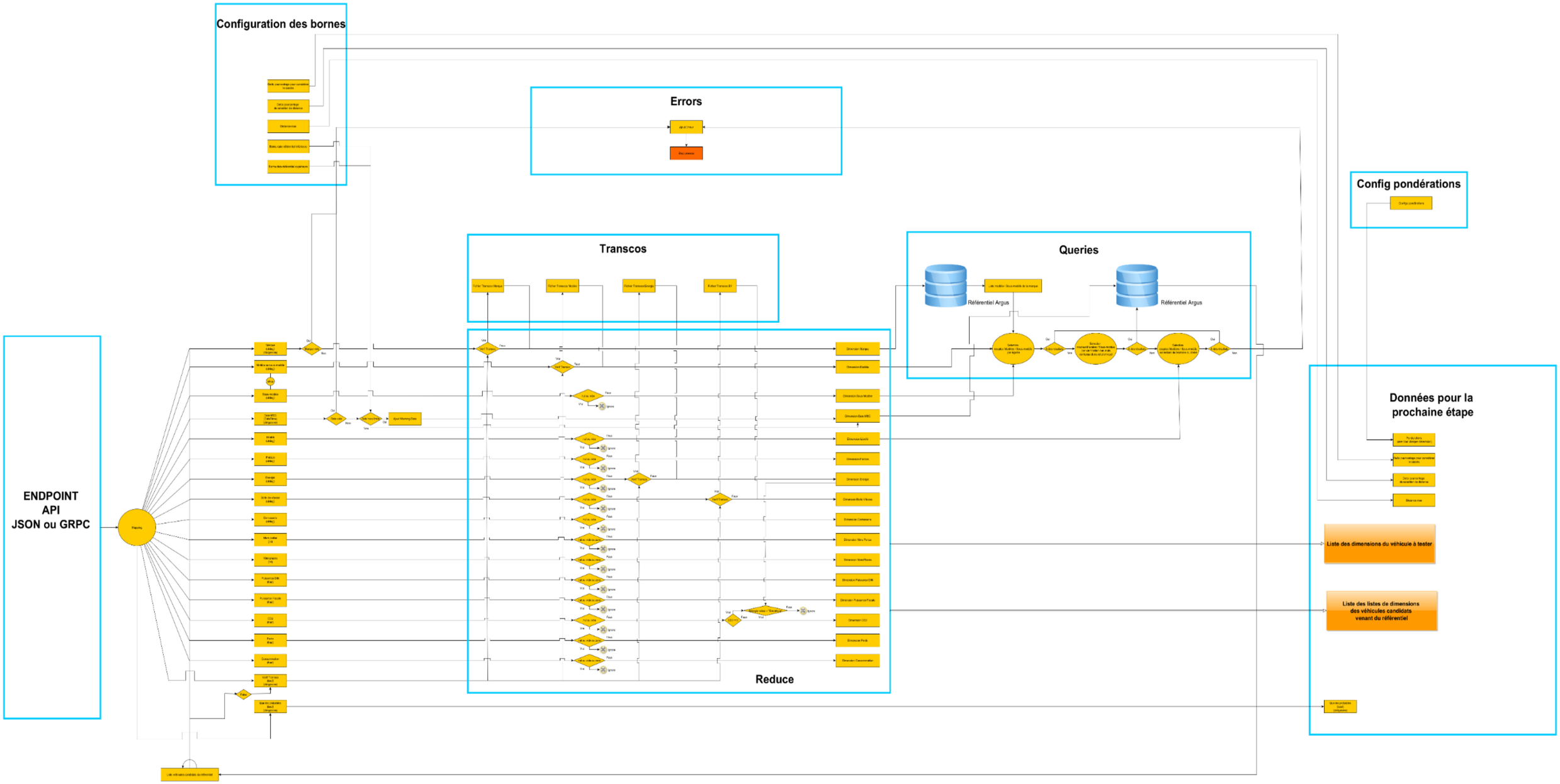
C'est également ici que l'on retrouve les fichiers .csv des différents types de transcodification.

3 : Schéma fonctionnel

Dans les prochaines parties, nous allons voir le fonctionnement au niveau du code des grandes étapes de l'algorithme. Dans un souci de synthèse, ce ne sera pas totalement exhaustif mais cela sera représentatif du parcours des données et de leur transformation. Je vais adopter, le plus possible, une approche chronologique, de l'arrivée des données dans le controller à la restitution de la réponse.

De plus, la prochaine partie contiendra un schéma fonctionnel de l'ensemble de l'algorithme, j'ai essayé d'être, en le concevant, le plus exhaustif possible tout en gardant au maximum la lisibilité. Il montre le parcours de (presque) chaque donnée de l'endpoint à la réponse.

3.1: Récupération des candidats au rapprochement



3.2: Calcul des distances

4 : Les controllers

4.1: L'injection de dépendance

Pour chacun des deux controllers, j'utilise l'injection de dépendances pour rendre disponible le contexte de la base de données ainsi que le système de gestion d'erreur dans toute l'application, et le mapping (librairie Automapper) dans le controller lui-même.

```
// DB Context
services.AddDbContext<CarlusContext>(options => options.UseNpgsql(settings.ConnectionStringReferentielModeles));

// Auto Mapper
var mapperConfig = new MapperConfiguration(config =>
{
    config.AddProfile<VehiculeProfile>();
});
mapperConfig.AssertConfigurationIsValid();

services.AddAutoMapper(typeof(Startup));

// Add Error System
services.AddScoped<IErrorsList, ErrorsList>();
```

Ci-dessus l'ajout des services pour l'injection de dépendances dans le Startup.

```
[Route("api/CalculerIdArgus")]
[ApiController]
1 reference | Caudron, 82 days ago | 3 authors, 28 changes
public class CalculerIdArgusController : ControllerBase
{
    private readonly CarlusContext _db;
    private readonly IMapper _mapper;
    private readonly IErrorsList _errorsList;

    0 references | Caudron, 189 days ago | 1 author, 1 change
    public CalculerIdArgusController(CarlusContext db, IMapper mapper, IErrorsList errorsList)
    {
        _db = db;
        _mapper = mapper;
        _errorsList = errorsList;
    }
```

Et l'on passe dans le constructeur les éléments que nous voulons injecter.

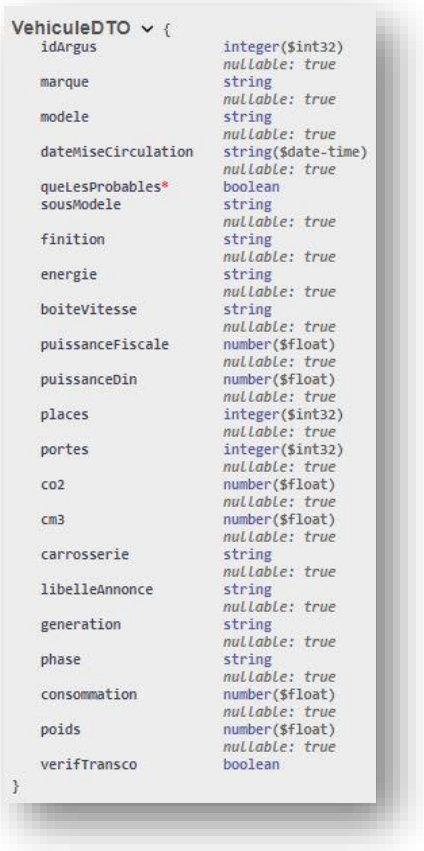
4.2: Le controller REST

```
/// <summary>
/// Calcul l'id Argus à partir de caractèristiques d'un véhicule
/// </summary>
/// <param name="vehiculeDTO">Caractéristiques du véhicule recherché</param>
/// <returns>Retourne la liste des véhicules les plus probables</returns>
[HttpPost]
[Authorize(AppSettings.API_PUBLIQUE)]
0 references | Caudron, 82 days ago | 2 authors, 13 changes
public async Task<ActionResult<CalculIdArgusDTO>> CalculerIdArgusAsync(VehiculeDTO vehiculeDTO)
{
    try
    {
        Vehicule vehicule = _mapper.Map<Vehicule>(vehiculeDTO);

        CalculerIdArgusService service = new CalculerIdArgusService(vehicule, _db, _errorsList);

        var result = await service.CalculAsync();

        var resultDto = _mapper.Map<CalculIdArgusDTO>(result);

        return resultDto;
    }
    catch
    {
        _errorsList.Add("Un problème général est survenu.", ErrorType.General);

        return new CalculIdArgusDTO
        {
            Succes = false,
            IdFound = "Pas de véhicule trouvé",
            Resultats = new List<ResultatIdArgusDTO>(),
            Errors = _mapper.Map<ErrorsListDTO>(_errorsList)
        };
    }
}
```

Concernant le controller REST en lui-même, le fonctionnement est très simple.

Il récupère donc en entrée un objet VehiculeDTO, qu'il mappe en Vehicule, instancie le service pour la recherche de l'id Argus, l'exécute, récupère la réponse et la mappe en son équivalent DTO.

Si jamais un gros problème non géré survient, nous renvoyons un DTO réponse vide avec l'ajout d'une erreur. Ce type de scénario est celui qui doit tendre à ne plus jamais exister.



Ci-contre le détail du VehiculeDTO que l'on a en entrée de l'API (représentation Swagger) avec les différentes propriétés ainsi que leurs types.

On peut remarquer que je demande l'idArgus en entrée nullable, c'est une façon d'activer en quelque sorte un mode « debug » de l'API. En effet, si l'on a déjà rapproché un idArgus auparavant, ce mode comparera les résultats précédents avec celui de mon API, et le cas échéant repérera le moment où il est perdu dans le cas où les résultats sont différents. C'est un mode pour lequel, par soucis de synthèse toujours, je n'aborderai pas beaucoup (d'ailleurs il n'apparait pas sur le schéma fonctionnel car il l'aurait trop complexifié).

De plus, il y a deux propriétés que je n'avais pas abordé : queLesProbables et verifTransco.

queLesProbables est un booléen qui, s'il est à true, fera en sorte de limiter les résultats renvoyés dans une certaine limite de distance (actuellement je renvoie tous les véhicules avec une distance comprise entre la distance minimale trouvée et la distance minimale + 10% de distance minimale).

verifTransco permet quant à lui de court circuiter la vérification des transcodifications si on le souhaite.

4.3: Le controller gRPC

```
3 references | 0 changes | 0 authors, 0 changes
public override async Task<CalculIdArgusResponse> CalculerIdArgus(VehiculeGrpc request, ServerCallContext context)
{
    try
    {
        Vehicule vehicule = _mapper.Map<Vehicule>(request);

        CalculerIdArgusService service = new CalculerIdArgusService(vehicule, _db, _errorsList);

        var result = await service.CalculAsync();

        var resultMapped = _mapper.Map<CalculIdArgusResponse>(result);

        return resultMapped;
    }
    catch
    {
        _errorsList.Add("Un problème général est survenu.", ErrorType.General);

        var result = new CalculIdArgusDTO
        {
            IdFound = "Pas de véhicule trouvé",
            Resultats = new List<ResultatIdArgusDTO>()
        };

        return _mapper.Map<CalculIdArgusResponse>(result);
    }
}
```

Le controller gRPC est très similaire à son homologue REST. La principale différence réside dans les types reçus et renvoyés, ainsi que l'appel du contexte du serveur.

Ces types sont définis par un fichier proto, qui déterminera la structure des données. Et grâce à un package Nuget, un code behind sera automatiquement généré pour effectuer la communication.

```
13  message VehiculeRequest {
14      google.protobuf.Int32Value idArgus = 1;
15      google.protobuf.StringValue marque = 2;
16      google.protobuf.StringValue modele = 3;
17      google.protobuf.Timestamp dateMec = 4;
18      bool queLesProbables = 5;
19      google.protobuf.StringValue sousModele = 6;
20      google.protobuf.StringValue finition = 7;
21      google.protobuf.StringValue energie = 8;
22      google.protobuf.StringValue boiteVitesse = 9;
23      google.protobuf.FloatValue puissanceFiscale = 10;
24      google.protobuf.FloatValue puissanceDin = 11;
25      google.protobuf.Int32Value nbPlaces = 12;
26      google.protobuf.Int32Value nbPortes = 13;
27      google.protobuf.FloatValue co2 = 14;
28      google.protobuf.FloatValue cm3 = 15;
29      google.protobuf.StringValue carrosserie = 16;
30      google.protobuf.StringValue libelleAnnonce = 17;
31      google.protobuf.StringValue generation = 18;
32      google.protobuf.StringValue phase = 19;
33      google.protobuf.FloatValue consommation = 20;
34      google.protobuf.FloatValue poids = 21;
35      bool verifTransco = 22;
36  }
```

A gauche : l'objet VehiculeRequest (équivalent du VehiculeDTO) en proto.

Ci-dessous : extrait du code généré automatiquement.

```
/// <summary>Field number for the "dateMec" field.</summary>
public const int DateMecFieldNumber = 4;
private global::Google.Protobuf.WellKnownTypes.Timestamp dateMec_;
[global::System.Diagnostics.DebuggerNonUserCodeAttribute]
11 references | 0 changes | 0 authors, 0 changes
public global::Google.Protobuf.WellKnownTypes.Timestamp DateMec
{
    get { return dateMec_; }
    set
    {
        dateMec_ = value;
    }
}
```

5 : Le système de gestion d'erreurs

Parce qu'il peut intervenir à peu près partout dans l'application, je vais aborder de suite le système de gestion des erreurs que j'ai développé afin d'avoir, dans les réponses, une traçabilité des différents soucis que nous pourrions rencontrer.

J'ai déjà évoqué dans le controller l'erreur générale particulière du plantage du code que l'on intercepte dans le catch.

```
namespace Errors
{
    23 references | Caudron, 115 days ago | 1 author, 5 changes
    public enum ErrorType
    {
        [Description("Erreur générale")]
        General,

        [Description("Erreur base de donnée")]
        Bdd,

        [Description("Erreur tri")]
        Tri,

        [Description("Transco")]
        Transco,

        [Description("Warning")]
        Warning
    }

    0 references | Caudron, 190 days ago | 1 author, 1 change
    public static class ErrorTypeExtensions
    {
        1 reference | Caudron, 190 days ago | 1 author, 1 change
        public static string EnumToDescription(this ErrorType value)
        {
            DescriptionAttribute[] attributes =
                (DescriptionAttribute[])value
                    .GetType()
                    .GetField(value.ToString())
                    .GetCustomAttributes(typeof(DescriptionAttribute), false);

            return attributes.Length > 0 ? attributes[0].Description : string.Empty;
        }
    }
}
```

Tout d'abord j'ai défini les types d'erreurs dans une enum. Par ordre d'importance.

De plus, le système est couplé avec une classe statique qui servira à retrouver la description de l'enum en texte, ce qui sera bien plus lisible car une enum finalement ne représente qu'une valeur numérique.

La classe Error en elle-même contient donc un type, le message du type, et un message que l'on pourra personnaliser en fonction de ce qu'il s'est passé.

```
namespace Errors
{
    7 references | Caudron, 190 days ago | 1 author, 2 changes
    public class Error
    {
        2 references | Caudron, 195 days ago | 1 author, 1 change
        public ErrorType Type { get; set; }

        1 reference | Caudron, 190 days ago | 1 author, 1 change
        public string TypeMessage { get; set; }

        1 reference | Caudron, 195 days ago | 1 author, 1 change
        public string Message { get; set; }
    }
}
```

```
namespace Errors
{
    2 references | Caudron, 161 days ago | 1 author, 5 changes
    public class ErrorsList : IErrorsList
    {
        3 references | Caudron, 195 days ago | 1 author, 1 change
        public int ErrorNb { get; set; }
        5 references | Caudron, 195 days ago | 1 author, 1 change
        public List<Error> ErrorList { get; set; }

        0 references | Caudron, 190 days ago | 1 author, 2 changes
        public ErrorsList()
        {
            ErrorNb = 0;
            ErrorList = new List<Error>();
        }

        19 references | Caudron, 161 days ago | 1 author, 4 changes
        public void Add(string message, ErrorType type)
        {
            Error error = new Error
            {
                Message = message,
                Type = type,
                TypeMessage = type.EnumToDescription()
            };

            ErrorNb++;

            ErrorList.Add(error);

            ErrorList = ErrorList.OrderBy(x => x.Type).ToList();
        }
    }
}
```

Et enfin, le tout est géré par une dernière classe qui contient la liste des erreurs, le nombre d'erreurs, ainsi qu'une fonction que l'on appellera pour ajouter une erreur à la liste.

Cette fonction construit l'erreur en fonction des infos passées en paramètres et l'ajoute à la liste, incrémente le nombre d'erreurs, et enfin tries par type (donc par importance) la liste des erreurs.

Le tout implémentant une interface qui permettra, en particulier, l'injection de dépendance.

3 : Le service Calculer Id Argus

```
namespace Services.Implementations
{
    5 references | Caudron, 71 days ago | 3 authors, 58 changes
    public class CalculerIdArgusService : ICalculerIdArgusService
    {
        private readonly IVehicule _vehicule;
        private readonly CarlusContext _db;
        private readonly IErrorsList _errorsList;
        private float? _percentDiffPrix;
        private float? _percentDiffPrixDistanceEgales;

        2 references | Caudron, 190 days ago | 1 author, 1 change
        public CalculerIdArgusService(IVehicule vehicule, CarlusContext db, IErrorsList errorsList)
        {
            _vehicule = vehicule;
            _db = db;
            _errorsList = errorsList;
        }

        3 references | Caudron, 71 days ago | 2 authors, 29 changes
        public async Task<CalculIdArgus> CalculAsync()
        {
            //TODO : Prévoir grosse refacto du service!! Séparer les responsabilités!!

            if (_vehicule.DateMiseCirculation == new DateTime(1, 1, 1) || _vehicule.Marque == null)
            {
                return BreakIfNoDateMecOrMarque();
            }

            AddWarningDateMEC();

            ListDimensionsVehicule dimensionsVehiculeATester = _vehicule.Reduce(false);

            AddTranscoErrors(dimensionsVehiculeATester);

            // TODO : utiliser brighter (cf projet Queries)

            var getVehRef = new GetVehiculesRefMarqueModeleDateMecQueryHandler(dimensionsVehiculeATester, _db, _errorsList);
            var vehiculesReferentiel = await getVehRef.ExecuteAsync();

            if (HasNoResult(vehiculesReferentiel))
            {
                return BreakProcess();
            }

            List<ListDimensionsVehicule> dimensionsVehRef = ReduceVehReferentiel(vehiculesReferentiel);
```

Le service, appelé par les controllers, est comme dit précédemment le chef d'orchestre de l'application.

Il va déterminer la chronologie des opérations à effectuer, ainsi que le formatage de la réponse qu'il renverra au controller.

Il est composé d'une fonction principale (CalculAsync()), ainsi que, par refactorisation et lisibilité, d'une multitude de sous-fonctions.

Par exemple, dans la capture précédente, nous pouvons voir le début du processus. La première étape étant de vérifier si une marque et une date de MEC ont été renseignées. Si ce n'est pas le cas, on arrête à cette étape avec l'ajout d'une erreur.

```
/// <summary>
/// Arrete le traitement et ajoute une erreur s'il manque la date MEC ou la marque dans le véhicule à comparer
/// </summary>
/// <returns></returns>
1 reference | Caudron, 113 days ago | 1 author, 3 changes
private CalculIdArgus BreakIfNoDateMecOrMarque()
{
    if (_vehicule.DateMiseCirculation == new DateTime(1, 1, 1))
    {
        _errorsList.Add("Pas de date MEC", ErrorType.General);
        return BreakProcess();
    }
    if (_vehicule.Marque == "")
    {
        _errorsList.Add("Pas de marque renseignée", ErrorType.General);
        return BreakProcess();
    }

    return null;
}
```

La fonction BreakProcess() qui renvoie un résultat vide avec la liste d'erreurs trouvées.

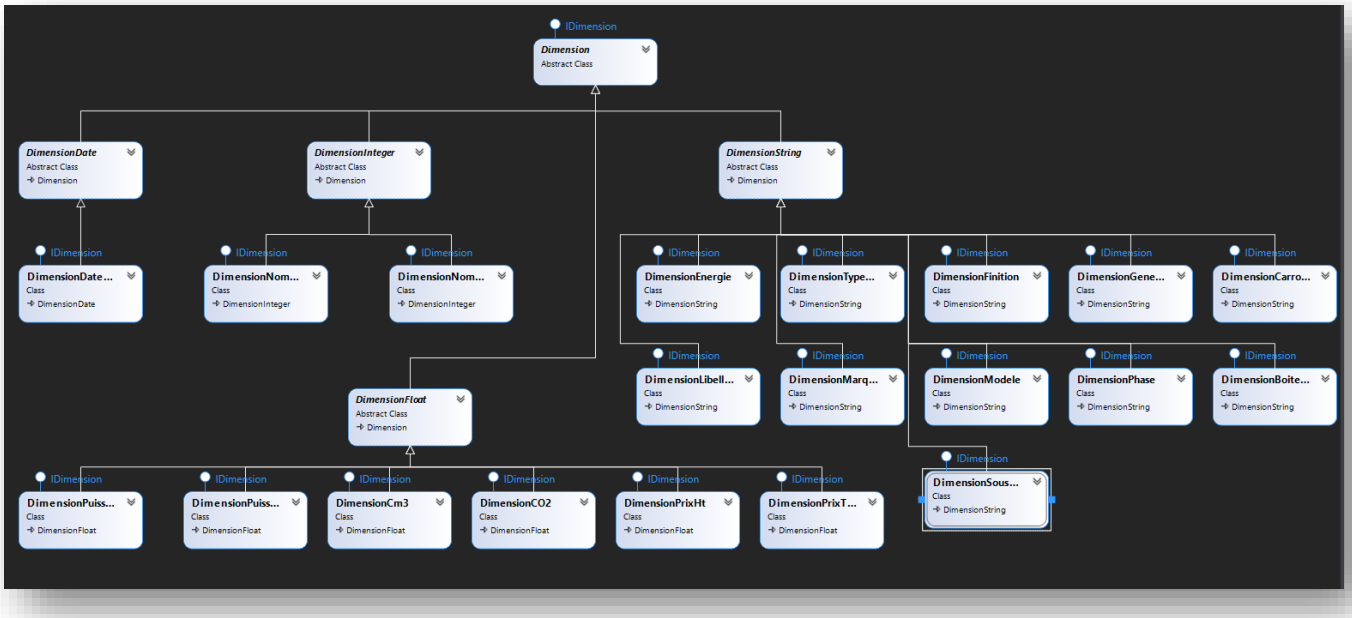
```
/// <summary>
/// Renvoi un résultat vide avec les erreurs
/// </summary>
/// <param name="vehiculesReferentiel"></param>
/// <returns></returns>
6 references | Caudron, 161 days ago | 1 author, 2 changes
private CalculIdArgus BreakProcess()
{
    _errorsList.Add("Pas de véhicule retourné", ErrorType.General);

    return new CalculIdArgus
    {
        IdFound = "Aucun véhicule trouvé",
        Errors = _errorsList
    };
}
```

Ensuite, il va ajouter des « Warning » dans la liste d'erreurs si la date est en dehors d'un intervalle que nous avons défini.

Et l'étape suivante, et non des moindre, est d'appeler la fonction Reduce() implémentée dans la classe Vehicule qui aura la responsabilité de transformer le véhicule à tester en liste de ses dimensions.

4 : La fonction Reduce() et les dimensions

7.1: Les dimensions



Les dimensions, comme on peut le constater avec le diagramme de classes ci-dessus, ont été organisées via héritage, dans le but de pouvoir hiérarchiser certains traitements. En effet, certains traitements sur des dimensions de même type (par exemple les chaines de caractères), seront effectués dans le niveau intermédiaire DimensionString, ce qui améliore la maintenabilité et évite la duplication du code.

```
14 references | Caudron, 116 days ago | 2 authors, 7 changes
public abstract class Dimension : IDimension
{
    internal object _value;

    1 reference | Cedric Caudron, 305 days ago | 1 author, 1 change
    public int IdArgus { get; set; }
    64 references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public object Value { get; set; }
    23 references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public float Ponderation { get; set; }
    99+ references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public float? DistanceDimension { get; set; }
    5 references | Cedric Caudron, 211 days ago | 1 author, 1 change
    public float? DistanceDimensionString { get; set; }
    5 references | Cedric Caudron, 211 days ago | 1 author, 1 change
    public float? DistanceDimensionStringLevenshtein { get; set; }
    13 references | Cedric Caudron, 284 days ago | 1 author, 1 change
    public List<TranscoResult> TranscoResults { get; set; }
    11 references | Cedric Caudron, 284 days ago | 1 author, 1 change
    public bool VerifTransco { get; set; }

    4 references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public Dimension()
    {
    }

    1 reference | Cedric Caudron, 305 days ago | 1 author, 1 change
    public object Convert()
    {
        return _value;
    }

    4 references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public Dimension(object Value) : base()
    {
        _value = Value;
    }

    54 references | Cedric Caudron, 262 days ago | 1 author, 1 change
    public virtual float? CalculerDistance(IDimension referentiel)
    {
        if (Value.Equals(referentiel.Value))
        {
            return 0;
        }
        else
        {
            return 1;
        }
    }
}
```

De plus, ils implémentent tous l'interface IDimension, ce qui sera important pour la suite en pouvant généraliser certains traitements qui ne seront par conséquent plus obligatoirement spécifiques à un type précis de dimension, mais qui pourra s'appliquer indifféremment à toutes.

Ci-contre la super classe Dimension représentant la plus haute strate de cette hiérarchie, avec ses fonctions Convert() et CalculerDistance(), dupliquée dans chaque strate, sur lesquelles je reviendrai ensuite.

```
29 references | Caudron, 79 days ago | 2 authors, 5 changes
public abstract class DimensionString : Dimension
{
    26 references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public new string Value { get; set; }

    11 references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public DimensionString(string Value) : base(Value)
    {
    }

    5 references | Cedric Caudron, 305 days ago | 1 author, 1 change
    protected DimensionString() : base()
    {
    }

    /// <summary>
    /// Traitements de conversions généraux pour toutes les dimensions Strings
    /// </summary>
    /// <returns>string sans caractères spéciaux, accents, espaces autour et en majuscules</returns>
    11 references | Cedric Caudron, 298 days ago | 1 author, 2 changes
    public new string Convert()
    {
        _value = UtilitairesStrings.EnleverEspacesAccentsEtMettreMajuscule(_value);

        return Value = (string)_value;
    }

    /// <summary>
    /// Calcul la distance entre deux chaines de caractères
    /// </summary>
    /// <param name="dimensionVehiculeRef">Dimension venant du référentiel</param>
    /// <returns>Valeur numérique de la distance entre les deux chaines</returns>
    29 references | Cedric Caudron, 262 days ago | 1 author, 1 change
    public override float? CalculerDistance(IDimension dimensionVehiculeRef)
    {
        DistanceDimension = base.CalculerDistance(dimensionVehiculeRef);

        return DistanceDimension;
    }
```

La classe intermédiaire DimensionString, avec sa méthode Convert() qui va faire un traitement valable pour toutes les dimensions de types chaines de caractères à la création de la Dimension. Dans ce cas, on nettoie la valeur en enlevant les caractères spéciaux, accents et espaces, et on met en majuscules grâce à une méthode du projet Utilitaires.

Et enfin, le niveau le plus bas de la hiérarchie, le seul qui puisse être instancié par ailleurs, avec ici pour exemple la classe DimensionMarque.

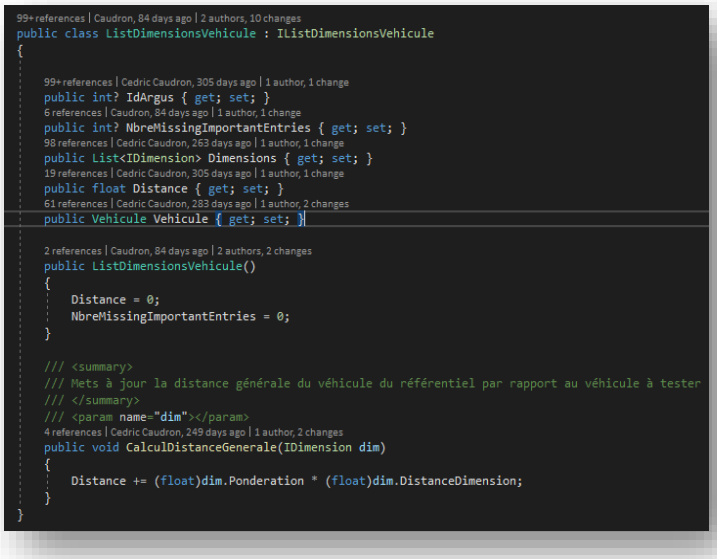
Ainsi, à chaque nouvelle instance de la DimensionMarque avec le paramètre verifTransco et la valeur en chaine de caractères, l'objet s'initialise en affectant la pondération qui lui est propre, et appelle la fonction Convert() afin d'appliquer les premiers traitements.

Dans ce cas, la fonction appelle son homologue de la classe parente vue juste au-dessus, puis c'est à ce niveau que nous appliquons les transcodifications.

```
5 references | Cedric Caudron, 246 days ago | 2 authors, 11 changes
public class DimensionMarque : DimensionString, IDimension
{
    0 references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public DimensionMarque() : base()
    {
    }

    1 reference | Cedric Caudron, 262 days ago | 1 author, 2 changes
    public DimensionMarque(string Value, bool verifTransco) : base(Value)
    {
        Ponderation = PonderationsConfig.Marque;
        VerifTransco = verifTransco;
        _ = Convert();
    }

    1 reference | Cedric Caudron, 246 days ago | 1 author, 8 changes
    public new string Convert()
    {
        try
        {
            _value = base.Convert();

            if (VerifTransco)
            {
                MarqueTranscos marqueTranscos = new MarqueTranscos((string)_value);
                GetTransco getTransco = new GetTransco(marqueTranscos);
                List<TranscoResult> resultTransco = getTransco.GetValue();
                if (resultTransco.Count != 0)
                {
                    _value = resultTransco[0].Value;
                }
            }

            RemonterValue(_value);

            return Value = (string)_value;
        }
        catch
        {
            return Value = string.Empty;
        };
    }
```

La dimension est donc créée avec toutes les informations nécessaires pour la suite : sa valeur nettoyée, transcodée le cas échéant et sa pondération. Sur certaines dimensions les transcodifications seront gardées sous forme de liste de plusieurs possibilités (dans le cas de la marque nous ne pouvons en avoir qu'une).

7.2: La classe ListDimensionsVehicule



```
99+ references | Caudron, 84 days ago | 2 authors, 10 changes
public class ListDimensionsVehicule : IListDimensionsVehicule
{

    99+ references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public int? IdArgus { get; set; }
    6 references | Caudron, 84 days ago | 1 author, 1 change
    public int? NbreMissingImportantEntries { get; set; }
    98 references | Cedric Caudron, 263 days ago | 1 author, 1 change
    public List<IDimension> Dimensions { get; set; }
    19 references | Cedric Caudron, 305 days ago | 1 author, 1 change
    public float Distance { get; set; }
    61 references | Cedric Caudron, 283 days ago | 1 author, 2 changes
    public Vehicule Vehicule { get; set; }

    2 references | Caudron, 84 days ago | 2 authors, 2 changes
    public ListDimensionsVehicule()
    {
        Distance = 0;
        NbreMissingImportantEntries = 0;
    }

    /// <summary>
    /// Mets à jour la distance générale du véhicule du référentiel par rapport au véhicule à tester
    /// </summary>
    /// <param name="dim"></param>
    4 references | Cedric Caudron, 249 days ago | 1 author, 2 changes
    public void CalculDistanceGenerale(IDimension dim)
    {
        Distance += (float)dim.Ponderation * (float)dim.DistanceDimension;
    }
}
```

Au niveau juste au-dessus de la dimension, cette classe va représenter un véhicule avec toutes les informations importantes de celui-ci. Les objets instanciés avec cette classe seront donc ceux sur lesquels nous travaillerons en majorité.

Il contient la liste des dimensions du véhicule, l'objet Vehicule d'origine et la distance générale quand il s'agira d'un véhicule extrait du référentiel candidat au rapprochement.

Lors de l'instanciation, on met simplement la distance à 0 (ainsi que l'autre propriété, qui représente le nombre de caractéristiques jugées importantes manquantes en entrée, mais je ne rentrerai pas dans le détail de ce traitement car il a surtout servi un temps pour l'analyse. Mais à terme il peut être un indice de fiabilité du résultat parmi d'autres).

C'est cette classe qui porte également le calcul de la distance générale.

7.3: La fonction Reduce()

Revenons donc dans notre service. Nous pouvons donc voir que nous récupérons un objet ListDimensionsVehicule en appelant la fonction Reduce() dans l'objet Vehicule lui-même.

C'est elle qui va créer toutes les dimensions, ainsi que l'objet ListDimensionsVehicule du véhicule à tester.

Elle prend en paramètre un booléen qui représente le fait que ce véhicule vient du référentiel ou non. Car les traitements ne seront pas tout à fait les mêmes suivant les cas.

```
/// <summary>
/// Transformer l'objet Véhicule en liste de ses différentes dimensions
/// </summary>
/// <returns>Retourne une liste de Dimension du véhicule</returns>
3 references | Caudron, 84 days ago | 1 author, 1 change
public ListDimensionsVehicule Reduce(bool isReferentiel)
{
    ListDimensionsVehicule caracteristiquesVehicule = new ListDimensionsVehicule
    {
        Vehicule = this,
        Dimensions = new List<IDimension>()
    };

    if (IdArgus != null)
    {
        caracteristiquesVehicule.IdArgus = IdArgus;
    }

    caracteristiquesVehicule.Dimensions.Add(new DimensionMarque(Marque, VerifTransco));
    caracteristiquesVehicule.Dimensions.Add(new DimensionModele(Modele, VerifTransco));
    caracteristiquesVehicule.Dimensions.Add(new DimensionDateMEC(DateMiseCirculation));

    if (!String.IsNullOrWhiteSpace(LibelleAnnonce))
    {
        caracteristiquesVehicule.Dimensions.Add(new DimensionLibelleAnnonce(LibelleAnnonce));
    }

    if (String.IsNullOrWhiteSpace(SousModele))
    {
        caracteristiquesVehicule.Dimensions.Add(new DimensionSousModele(Modele));
    }
    else
    {
        caracteristiquesVehicule.Dimensions.Add(new DimensionSousModele(SousModele));
    }

    if (!String.IsNullOrWhiteSpace(Energie))
    {
        caracteristiquesVehicule.Dimensions.Add(new DimensionEnergie(Energie, VerifTransco));
    }

    if (Co2 != null)
    {
        if (Co2 != 0 || (string)caracteristiquesVehicule.Dimensions.Where(d => d is DimensionEnergie).Select(v => v.Value).FirstOrDefault() == "Electrique")
        {
            caracteristiquesVehicule.Dimensions.Add(new DimensionCO2((float)Co2));
        }
    }
```

Extrait du code de la fonction Reduce() dans la classe Vehicule.

Pour chaque caractéristique de véhicule, si elle existe, j'instancies et ajoute à la liste des dimensions la dimension correspondante.

A noter ici la subtilité concernant le CO2. En effet, je ne rajoute une dimension pour cette caractéristique si elle est égale à zéro uniquement si la dimension énergie existe et a pour valeur « Electrique ». D'où ici l'importance de l'ordre dans la création des dimensions.

A noter également que certaines dimensions prennent en paramètre VerifTransco alors que d'autres non. Sur ce paramètre, la plupart du temps il sera vrai pour les véhicules en entrées, mais il sera toujours faux pour les véhicules du référentiel. En effet, le but des transcodifications étant justement de trouver des correspondances avec les données du référentiel, cela n'a aucun sens de les vérifier sur les véhicules venant dudit référentiel. Sans compter la perte de performances que cela engendrerait.

```
if (PuissanceDin != null && PuissanceDin != 0)
{
    caracteristiquesVehicule.Dimensions.Add(new DimensionPuissanceDin((float)PuissanceDin));
}
else
{
    if (isReferentiel)
    {
        caracteristiquesVehicule.Dimensions.Add(new DimensionPuissanceDin(0F));
    }
}
```

Concernant le paramètre isReferentiel évoqué plus haut, il sert à forcer la création d'une dimension puissance Din dans le cas où dans le référentiel l'information n'y est pas, ce qui arrive parfois, et au vu des analyses sur les résultats, et que cette dimension est apparue comme très importante, cela force à la prendre toujours en compte.

A ce stade, nous avons donc notre objet ListDimensionsVehicule qui correspond au véhicule envoyé dans l'API et donc dont nous cherchons la correspondance, prêt à être utilisé.

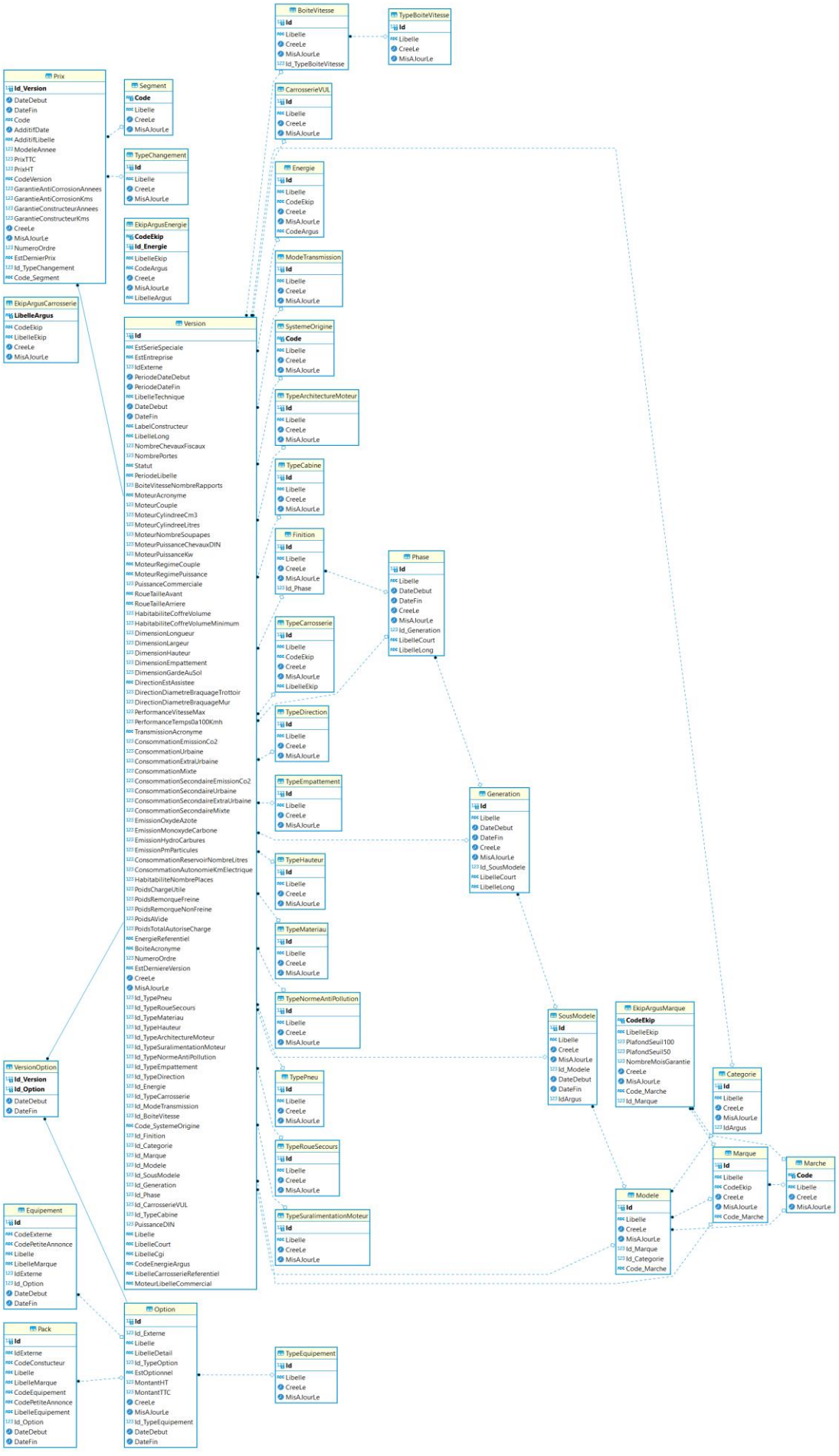
5 : La récupération des candidats potentiels

8.1: Le référentiel Argus

Dans l'environnement de l'entreprise, le référentiel Argus est intégré dans une base relationnelle PostgreSQL, ainsi j'ai utilisé la même intégration au niveau local, sachant, qu'à terme, toutes les interactions avec celui-ci seront externalisées dans une API indépendante.

Page suivante : le diagramme du référentiel Argus tel que je l'ai utilisé.

On peut y remarquer que la table Version est celle qui portera la majorité des caractéristiques des véhicules, dont le fameux id Argus qui est l'information que l'on cherche (IdExterne de la table Version).


BoiteVitesse
Id
Libelle
CreeLe
MisAJourLe
Id_TypeBoiteVitesse
TypeBoiteVitesse
Id
Libelle
CreeLe
MisAJourLe
Prix
Id_Version
DateDebut
DateFin
Code
AdditifDate
AdditifLibelle
ModeleAnnee
PrixTTC
PrixHT
CodeVersion
GarantieAntiCorrosionAnnees
GarantieAntiCorrosionKms
GarantieConstructeurAnnees
GarantieConstructeurKms
CreeLe
MisAJourLe
NumeroOrdre
EstDernierPrix
Id_TypeChangement
Code_Segment
Segment
Code
Libelle
CreeLe
MisAJourLe
TypeChangement
Id
Libelle
CreeLe
MisAJourLe
EkipArgusEnergie
CodeEkip
Id_Energie
LibelleEkip
CodeArgus
CreeLe
MisAJourLe
LibelleArgus
EkipArgusCarrosserie
LibelleArgus
CodeEkip
LibelleEkip
CreeLe
MisAJourLe
CarrosserieVUL
Id
Libelle
CreeLe
MisAJourLe
Energie
Id
Libelle
CodeEkip
CreeLe
MisAJourLe
CodeArgus
ModeTransmission
Id
Libelle
CreeLe
MisAJourLe
SystemeOrigine
Code
Libelle
CreeLe
MisAJourLe
TypeArchitectureMoteur
Id
Libelle
CreeLe
MisAJourLe
TypeCabine
Id
Libelle
CreeLe
MisAJourLe
Finition
Id
Libelle
CreeLe
MisAJourLe
Id_Phase
Phase
Id
Libelle
DateDebut
DateFin
CreeLe
MisAJourLe
Id_Generation
LibelleCourt
LibelleLong
TypeCarrosserie
Id
Libelle
CodeEkip
CreeLe
MisAJourLe
LibelleEkip
TypeDirection
Id
Libelle
CreeLe
MisAJourLe
TypeEmpattement
Id
Libelle
CreeLe
MisAJourLe
Generation
Id
Libelle
DateDebut
DateFin
CreeLe
MisAJourLe
Id_SousModele
LibelleCourt
LibelleLong
TypeHauteur
Id
Libelle
CreeLe
MisAJourLe
TypeMateriau
Id
Libelle
CreeLe
MisAJourLe
TypeNormeAntiPollution
Id
Libelle
CreeLe
MisAJourLe
SousModele
Id
Libelle
CreeLe
MisAJourLe
Id_Modele
DateDebut
DateFin
IdArgus
EkipArgusMarque
CodeEkip
LibelleEkip
PlafondSeuil100
PlafondSeuil50
NombreMoisGarantie
CreeLe
MisAJourLe
Code_Marche
Id_Marque
Categorie
Id
Libelle
CreeLe
MisAJourLe
IdArgus
Marque
Id
Libelle
CodeEkip
CreeLe
MisAJourLe
Code_Marche
Marche
Code
Libelle
CreeLe
MisAJourLe
Modele
Id
Libelle
CreeLe
MisAJourLe
Id_Marque
Id_Categorie
Code_Marche
TypePneu
Id
Libelle
CreeLe
MisAJourLe
TypeRoueSecours
Id
Libelle
CreeLe
MisAJourLe
TypeSuralimentationMoteur
Id
Libelle
CreeLe
MisAJourLe
Version
Id
EstSerieSpeciale
EstEntreprise
IdExterne
PeriodeDateDebut
PeriodeDateFin
LibelleTechnique
DateDebut
DateFin
LabelConstructeur
LibelleLong
NombreChevauxFiscaux
NombrePortes
Statut
PeriodeLibelle
BoiteVitesseNombreRapports
MoteurAcronyme
MoteurCouple
MoteurCylindreeCm3
MoteurCylindreeLitres
MoteurNombreSoupapes
MoteurPuissanceChevauxDIN
MoteurPuissanceKw
MoteurRegimeCouple
MoteurRegimePuissance
PuissanceCommerciale
RoueTailleAvant
RoueTailleArriere
HabitabiliteCoffreVolume
HabitabiliteCoffreVolumeMinimum
DimensionLongueur
DimensionLargeur
DimensionHauteur
DimensionEmpattement
DimensionGardeAuSol
DirectionEstAssistee
DirectionDiametreBraquageTrottoir
DirectionDiametreBraquageMur
PerformanceVitesseMax
PerformanceTempsda100Kmh
TransmissionAcronyme
ConsommationEmissionCo2
ConsommationUrbaine
ConsommationExtraUrbaine
ConsommationMixte
ConsommationSecondaireEmissionCo2
ConsommationSecondaireUrbaine
ConsommationSecondaireExtraUrbaine
ConsommationSecondaireMixte
EmissionOxydeAzote
EmissionMonoxydeCarbone
EmissionHydroCarbures
EmissionPmParticules
ConsommationReservoirNombreLitres
ConsommationAutonomieKmElectrique
HabitabiliteNombrePlaces
PoidsChargeUtile
PoidsRemorqueFreine
PoidsRemorqueNonFreine
PoidsAVide
PoidsTotalAutoriseCharge
EnergieReferentiel
BoiteAcronyme
NumeroOrdre
EstDerniereVersion
CreeLe
MisAJourLe
Id_TypePneu
Id_TypeRoueSecours
Id_TypeMateriau
Id_TypeHauteur
Id_TypeArchitectureMoteur
Id_TypeSuralimentationMoteur
Id_TypeNormeAntiPollution
Id_TypeEmpattement
Id_TypeDirection
Id_Energie
Id_TypeCarrosserie
Id_ModeTransmission
Id_BoiteVitesse
Code_SystemeOrigine
Id_Finition
Id_Categorie
Id_Marque
Id_Modele
Id_SousModele
Id_Generation
Id_Phase
Id_CarrosserieVUL
Id_TypeCabine
PuissanceDIN
Libelle
LibelleCourt
LibelleCgi
CodeEnergieArgus
LibelleCarrosserieReferentiel
MoteurLibelleCommercial
VersionOption
Id_Version
Id_Option
DateDebut
DateFin
Equipement
Id
CodeExterne
CodePetiteAnnonce
Libelle
LibelleMarque
IdExterne
Id_Option
DateDebut
DateFin
Option
Id
Id_Externe
Libelle
LibelleDetail
Id_TypeOption
EstOptionnel
MontantHT
MontantTTC
CreeLe
MisAJourLe
Id_TypeEquipement
DateDebut
DateFin
Pack
Id
IdExterne
CodeConstucteur
Libelle
LibelleMarque
CodeEquipement
CodePetiteAnnonce
LibelleEquipement
Id_Option
DateDebut
DateFin
TypeEquipement
Id
Libelle
CreeLe
MisAJourLe

8.2: Recherche des modèles/sous-modèles correspondants

```
/// <summary>
/// Recherche la liste des modèles selon la marque
/// </summary>
/// <returns>Liste de modèle de véhicule possibles</returns>
1 reference | Caudron, 210 days ago | 1 author, 1 change
private async Task<List<VehiculeModele>> GetModeleVehiculeByMarqueAsync()
{
    var query =
        from version in _db.Version
        join marque in _db.Marque on version.IdMarque equals marque.Id
        join modele in _db.Modele on version.IdModele equals modele.Id
        join sousModele in _db.SousModele on version.IdSousModele equals sousModele.Id
        where marque.Libelle == _marque
        select new VehiculeModele
        {
            IdModele = version.IdModele,
            LibelleModele = modele.Libelle,
            IdSousModele = sousModele.Id,
            LibelleSousModele = sousModele.Libelle
        };

    List<VehiculeModele> result = await query.Distinct().ToListAsync();

    return result;
}
```

La première étape va consister, à partir de la marque, de trouver dans le référentiel tous les modèles et sous-modèles de celle-ci, et ensuite de sélectionner le ou les candidats en fonction du modèle/sous-modèle envoyé, et, si le résultat n'est pas probant, de tenter de l'extraire du libellé.

Ci-dessous, la méthode gérant la sélection des modèles et sous-modèles à partir du libellé (donc si aucune correspondance trouvée avec celui en entrée, ou si tout simplement il n'y en a pas en entrée).

Je privilégie toujours la sélection du sous-modèle, car étant par définition plus précis que le modèle, afin que s'il est trouvé, la liste des candidats soit réduite automatiquement.

La troisième condition est une recherche du modèle parmi les mots séparés du libellé (car il arrive que les informations soient dans le désordre quelque fois).

```
/// <summary>
/// Cherche le modèle dans le libellé envoyé
/// </summary>
/// <param name="listModelesATrier"></param>
/// <param name="libelleATester"></param>
/// <returns></returns>
1 reference | Caudron, 130 days ago | 1 author, 4 changes
private List<VehiculeModele> VerifierSiModeleDansLibelle(List<VehiculeModele> listModelesATrier, string libelleATester)
{
    var result = new List<VehiculeModele>();

    foreach (VehiculeModele veh in listModelesATrier)
    {
        string cleanedLibelle = EnleverCaracteresSpeciauxEtPointsVirgules(libelleATester);
        cleanedLibelle = EnleverEspacesAccentsEtMettreMajuscule(cleanedLibelle);
        cleanedLibelle = EnleverEspacesInterieurs(cleanedLibelle);

        if (veh.LibelleSousModele.Length > 3 && cleanedLibelle.Contains(EnleverEspacesInterieurs(EnleverEspacesAccentsEtMettreMajuscule(veh.LibelleSousModele))))
        {
            result.Add(veh);
        }
        else if (veh.LibelleModele.Length > 3 && cleanedLibelle.Contains(EnleverEspacesAccentsEtMettreMajuscule(veh.LibelleModele)))
        {
            result.Add(veh);
        }
        else if (veh.LibelleModele.Length < 3 && IsModeleInLibelleFractionne(veh.LibelleModele, libelleATester))
        {
            result.Add(veh);
        }
    }

    return result;
}
```

8.3: Recherche des véhicules candidats

Nous avons donc maintenant une marque et la liste des modèles et/ou sous-modèles possibles pour le véhicule recherché. Il reste à trouver les véhicules qui correspondent à ces critères.

```
/// Cherche les candidats au rapprochement en fonction de la marque, date MEC et modèles trouvés auparavant
/// </summary>
/// <param name="modeleVehicule"></param>
/// <returns>Liste des véhicules candidats au rapprochement</returns>
1 reference | Caudron, 130 days ago | 2 authors, 11 changes
private async Task<List<Vehicule>> GetVehiculeByModeleDateAsync(List<VehiculeModele> modelesVehicule)
{
    List<Vehicule> result = new List<Vehicule>();

    var query =
    from version in _db.Version
    join marque in _db.Marque on version.IdMarque equals marque.Id
    join modele in _db.Modele on version.IdModele equals modele.Id
    join sousModele in _db.SousModele on version.IdSousModele equals sousModele.Id
    join energie in _db.Energie on version.IdEnergie equals energie.Id
    join prix in _db.Prix on version.Id equals prix.IdVersion
    join boiteVitesse in _db.BoiteVitesse on version.IdBoiteVitesse equals boiteVitesse.Id
    join typeBoiteVitesse in _db.TypeBoiteVitesse on boiteVitesse.IdTypeBoiteVitesse equals typeBoiteVitesse.Id
    join finition in _db.Finition on version.IdFinition equals finition.Id
    join generation in _db.Generation on version.IdGeneration equals generation.Id
    join phase in _db.Phase on version.IdPhase equals phase.Id
    where modelesVehicule.Select(v => v.IdModele).Contains(modele.Id) && modelesVehicule.Select(v => v.IdSousModele).Contains(sousModele.Id)
    where (version.DateDebut <= _dateMEC && version.DateFin >= _dateMEC.AddMonths(-12)) || (version.DateDebut <= _dateMEC && version.DateFin == null)
    select new Vehicule
    {
        IdArgus = (int?)version.IdExterne,
        Marque = marque.Libelle,
        Modele = modele.Libelle,
        SousModele = sousModele.Libelle,
        DateMiseCirculation = _dateMEC,
        DateDebutPeriode = version.PeriodeDateDebut,
        DateFinPeriode = version.PeriodeDateFin,
        DateDebut = version.DateDebut,
        DateFin = version.DateFin,
        Energie = energie.Libelle,
        BoiteVitesse = boiteVitesse.Libelle,
        TypeBoiteVitesse = typeBoiteVitesse.Libelle,
        PuissanceFiscale = version.NombreChevauxFiscaux,
        PuissanceDin = version.PuissanceDin,
        NbPortes = version.NombrePortes,
        NbPlaces = version.HabitabiliteNombrePlaces,
        Co2 = (float?)version.ConsommationEmissionCo2,
        Cm3 = version.MoteurCylindreeCm3,
        Carrosserie = version.LibelleCarrosserieReferentiel,
        Generation = generation.Libelle,
        Phase = phase.Libelle,
        LibelleAnnonce = version.LibelleLong,
        PrixHt = (float?)prix.PrixHt,
        PrixTTC = (float?)prix.PrixTtc,
        Poids = (float?)version.PoidsTotalAutoriseCharge,
        Consommation = (float?)version.ConsommationMixte,
        Finition = finition.Libelle,
        VerifTransco = false
    };

    return await query.ToListAsync();
}
```

On recherche les véhicules correspondants aux critères définis auparavant, et avec la date de Mise En Circulation (date MEC) à laquelle on applique une fenêtre de 12 mois, car le véhicule a pu être vendu après la fin officielle de la commercialisation par la marque (la médiane se trouvant aux alentours de 10 mois).

Nous avons donc ici nos véhicules sur lesquels nous allons pouvoir travailler. A noter que dans l'objet Vehicule issus du référentiel, je mets VerifTransco à false pour tous. Car nous allons réutiliser la fonction Reduce pour chacun de ces véhicules, et il est inutile, et contre-performant, de refaire le traitement des transcodifications dessus.

6 : Le traitement du libellé

Le libellé, comme j'en ai déjà parlé, porte souvent en lui de nombreuses informations qui relèvent de la responsabilité des différentes dimensions. Ainsi, il va avoir un traitement à part, comportant l'extraction de ces différentes informations, la création de leur dimensions respectives (si elles ne sont pas en entrée) et le nettoyages de ces mêmes informations que ce soit du coté du véhicule à tester, que des candidats venus du référentiel (nettoyage intervenant également si les données sont en entrées). Ceci a pour but de ne pas multiplier les mêmes traitements sur les mêmes données, et d'être le plus précis dans la récupération du véhicule.

Par exemple, si le nombre de portes figure dans le libellé en entrée mais pas dans celui du référentiel, il est plus logique de le traiter dans sa dimension propre avec les informations nombre de portes du référentiel.

Je ne vais pas trop entrer dans les détails de ces traitements qui sont particulièrement nombreux et complexes (de plus c'est une partie qui nécessite une importante refactorisation), et ce toujours par soucis de synthèse car sinon ce dossier prendrait une ampleur bien trop importante, mais je vais donner quelques exemples de code de certains de ces différents traitements.

```
/// <summary>
/// Ajout de dimension nombre de porte à partir du libellé
/// </summary>
1 reference | Caudron, 172 days ago | 2 authors, 4 changes
private void AddDimensionNbrePortes()
{
    if (!_dimensionsVehiculeATester.Dimensions.OfType<DimensionNombrePortes>().Any())
    {
        List<string> patterns = new List<string> { @"[1-7]\s?(PORTES|PORTE|P)" };
        List<string> wordsToReplace = new List<string> { "PORTES", " PORTES", "PORTE", " PORTE", "P", " P" };

        var nbrePortes = MatchingDimensionInLibelle(patterns);

        if (nbrePortes != "")
        {
            nbrePortes = CleanInFoDimension(wordsToReplace, nbrePortes);

            _dimensionsVehiculeATester.Dimensions.Add(new DimensionNombrePortes(int.Parse(nbrePortes)));
        }
    }
}
```

Ici la détection grâce à des Regex du nombre de portes dans le libellé, l'extraction de sa valeur et la création de sa dimension dans le véhicule à tester.

Ci-contre, le « moteur » servant à trouver les dimensions existantes et envoyer vers le nettoyage des informations du libellé, ci-dessous le nettoyage du nombre de portes.

```
1 reference | Caudron, 182 days ago | 2 authors, 3 changes
private ListDimensionsVehicule SuppressNbrePortesInLibelle(IDimension dim, ListDimensionsVehicule veh)
{
    var substrToSuppress = new List<string>
    {
        dim.Value.ToString() + "PORTES",
        dim.Value.ToString() + " PORTES",
        dim.Value.ToString() + "P",
        dim.Value.ToString() + " P"
    };

    foreach (var substr in substrToSuppress)
    {
        veh = CleanUpLibelleDimension.SuppressWordInLibelle(veh, substr);
    }

    return veh;
}
```

```
/// <summary>
/// Gère le nettoyage du libellé en cherchant les bonnes dimensions
/// </summary>
/// <param name="veh"></param>
/// <returns>La liste des dimensions du véhicule après nettoyage</returns>
2 references | Caudron, 83 days ago | 2 authors, 6 changes
private ListDimensionsVehicule CleanLibelle(ListDimensionsVehicule veh)
{
    foreach (var dim in veh.Dimensions)
    {
        switch (dim)
        {
            case DimensionMarque _:
            case DimensionModele _:
                veh = SuppressInLibelle(dim, veh);
                break;

            case DimensionNombrePlaces _:
                veh = SuppressNbrePlacesInLibelle(dim, veh);
                break;

            case DimensionPuissanceDin _:
                veh = SuppressPuissanceInLibelle(dim, veh);
                break;

            case DimensionNombrePortes _:
                veh = SuppressNbrePortesInLibelle(dim, veh);
                break;

            case DimensionGeneration _:
                veh = SuppressGenerationInLibelle(dim, veh);
                break;

            case DimensionPhase _:
                veh = SuppressPhaseInLibelle(dim, veh);
                break;

            default:
                break;
        }
    }

    return veh;
}
```

```
/// <summary>
/// Vérifie si la finition est présente dans le libellé à tester
/// </summary>
/// <param name="dimVehATester"></param>
/// <param name="finition"></param>
/// <returns>Booléen si la finition présente</returns>
1 reference | Caudron, 83 days ago | 2 authors, 5 changes
private bool FinitionIsInVehicule(ListDimensionsVehicule dimVehATester, string finition)
{
    string libelleATester = dimVehATester.Dimensions
        .Where(dim => dim is DimensionLibelleAnnonce)
        .Select(dim => (string)dim.Value).FirstOrDefault();

    libelleATester = UtilitairesStrings.EnleverCaracteresSpeciauxEtPointsVirgules(libelleATester);
    libelleATester = UtilitairesStrings.EnleverTirets(libelleATester);

    string finitionCleaned = UtilitairesStrings.EnleverTirets(finition);
    finitionCleaned = UtilitairesStrings.EnleverCaracteresSpeciauxEtPointsVirgules(finitionCleaned);

    if (IsFintitionNoSpaceIsInLibelleNoSpace(libelleATester, finitionCleaned))
    {
        return true;
    }
    else if (IsWordsOfFinitionAreInVehicule(libelleATester, finitionCleaned))
    {
        return true;
    }
    else if (IsFinitionInVehiculeWithTransco(libelleATester, finitionCleaned))
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

La finition dans le libellé a un des traitement les plus complexe de l'algorithme, car il existe de nombreux cas dû à la qualité de données en entrée qui peuvent provoquer des effets de bords.

Ci-contre, la méthode qui vérifie si la finition est bien présente dans le libellé. Avec trois traitements par ordre d'importance.

Le premier, vérifier si la finition sans espace est bien dans le libellé sans espace. Le deuxième, vérifier si les mots de la finition sont bien dans le libellé (dans le cas où ils seraient dans le désordre), et enfin le troisième qui vérifie si la finition est bien présente dans le libellé après transcodification.

Ci-contre le deuxième traitement, vérifiant que tous les mots de la finition se trouvent dans le libellé. Si un mot n'est pas présent, on qui le processus avec un résultat négatif.

```
2 references | Caudron, 202 days ago | 2 authors, 3 changes
private bool IsWordsOfFinitionAreInVehicule(string libelleATester, string finition)
{
    string[] libelleSplit = libelleATester.Split(' ');
    string[] finitionSplit = finition.Split(' ');
    bool result = false;

    foreach (string substr in finitionSplit)
    {
        foreach (string substrLibelle in libelleSplit)
        {
            if ((substr == substrLibelle) && (substr.Length > 2))
            {
                result = true;
                break;
            }
        }

        if (!result)
        {
            break;
        }
    }

    return result;
}
```

De plus chaque mot doit avoir une longueur de plus de 2 caractères pour éviter les effets de bord.

```
1 reference | Caudron, 182 days ago | 2 authors, 3 changes
private bool IsFinitionInVehiculeWithTransco(string libelleATester, string finition)
{
    FinitionsTranscos finitionsTranscos = new FinitionsTranscos(finition);
    GetTransco getTransco = new GetTransco(finitionsTranscos);
    List<TranscoResult> resultTransco = getTransco.GetValue();

    if (resultTransco.Count != 0)
    {
        foreach (TranscoResult transco in resultTransco)
        {
            if (libelleATester.Contains(UtilitairesStrings.EnleverTirets(transco.Value)))
            {
                return true;
            }
            else
            {
                libelleATester = libelleATester.Replace(transco.Value, transco.Key);

                if (IsWordsOfFinitionAreInVehicule(libelleATester, finition))
                {
                    return true;
                }
            }
        }

        return false;
    }
    else
    {
        return false;
    }
}
```

Et ici le troisième traitement qui vérifie avec les transcodifications si la finition transcodée est dans le libellé.

A noter l'utilisation du traitement précédent après la transcodification.

Il existe de nombreux autres traitements pour le libellé mais nous en avons vu une partie des principaux. Le but étant toujours de garnir des dimensions avec les informations du libellé.

7 : Le calcul des distances

10.1: Le matching entre les dimensions à comparer

Nous avons donc toutes nos dimensions, nettoyées et prêtes à être comparées, que ce soit du coté du véhicule à tester que celui des véhicules candidats. Il nous reste donc à calculer les distances et effectuer un classement des véhicules de la distance la plus faible à la plus élevée, ainsi que d'éliminer ceux que nous estimons trop éloignés pour être corrects.

La première étape (ci-dessous) est de faire matcher les dimensions de même type afin de comparer les mêmes données dans celui recherché et dans les candidats. C'est le moteur de base du calcul de distance.

```
/// <summary>
/// Parcours les différentes dimensions et trouve celles du même type dans les listes de dimensions des véhicules du référentiel et celui à tester
/// </summary>
/// <param name="dimensionsVehiculeATester">Liste dimensions véhicule à tester</param>
/// <param name="dimensionsVehRef">Liste de liste de dimensions des véhicules réucpérés dans le référentiel</param>
1 reference | Cedric Caudron, 260 days ago | 1 author, 1 change
private void MatchingDimensionsAndCalculDistance(ListDimensionsVehicule dimensionsVehiculeATester, List<ListDimensionsVehicule> dimensionsVehRef)
{
    foreach (ListDimensionsVehicule vehDimensionRef in dimensionsVehRef)
    {
        foreach (IDimension dimRef in vehDimensionRef.Dimensions)
        {
            foreach (var dimTest in dimensionsVehiculeATester.Dimensions.Where(dimTest => dimRef.GetType() == dimTest.GetType()))
            {
                ChoisirTraitementsSuivantLesDimensions(vehDimensionRef, dimRef, dimTest);
                break;
            }
        }
    }
}
```

A chaque fois qu'un type de dimension matche avec son homologue, par soucis de performances, je stoppe la boucle, car il ne peut (sauf problème) n'y avoir qu'un seul type de dimension dans chaque véhicule.

De plus, lorsque le type matche, on choisit un traitement spécifique suivant le type de dimension. En effet, il peut y avoir des spécificités suivant le type dans le traitement et c'est ce qui est géré dans l'image suivante.

Dans ce traitement, pour les cas des marques et modèles, nous ne faisons rien ce qui est logique vu que la marque et le modèle doivent correspondre vu qu'ils ont servi de base à la requête initiale.

```
/// <summary>
/// Selectionne les traitements à faire suivant les différentes dimensions
/// </summary>
/// <param name="vehDimensionRef"></param>
/// <param name="dimRef"></param>
/// <param name="dimTest"></param>
1 reference | Caudron, 82 days ago | 2 authors, 12 changes
private static void ChoisirTraitementsSuivantLesDimensions(ListDimensionsVehicule vehDimensionRef, IDimension dimRef, IDimension dimTest)
{
    switch (dimRef)
    {
        case DimensionModele _:
        case DimensionSousModele _:
            break;

        case DimensionFinition _:
            dimRef = CalcDistFinitionIfNoExist(dimRef, dimTest);
            vehDimensionRef.CalculDistanceGenerale(dimRef);
            break;

        case DimensionDateMEC _:
            vehDimensionRef.CalculDistanceGenerale(dimRef);
            break;

        default:
            if (dimTest.Value != null)
            {
                dimRef.CalculerDistance(dimTest);
            }
            else
            {
                dimRef.DistanceDimension = 1;
            }
            vehDimensionRef.CalculDistanceGenerale(dimRef);
            break;
    }
}
```

Dans la plupart des cas, je fais appel à la fonction CalculerDistance() de la dimension du référentiel (car ce sont les candidats qui porteront leur propre distance). Cette fonction étant spécifique à chaque type de dimension suivant les méthodes de calcul définies dans la partie théorique.

Et enfin la distance générale du véhicule est mise à jour avec la fonction CalculDistanceGenerale().

10.2: Le calcul de base des distances

```
54 references | Cedric Caudron, 273 days ago | 1 author, 1 change
public virtual float? CalculerDistance(IDimension referentiel)
{
    if (Value.Equals(referentiel.Value))
    {
        return 0;
    }
    else
    {
        return 1;
    }
}
```

Chaque calcul de distance passera par appeler la fonction dans la super classe Dimension. Qui vérifiera l'égalité parfaite. Et attribuera une distance de 0 ou de 1 en fonction du résultat.

10.3: Distance entre deux entiers

```
12 references | Cedric Caudron, 273 days ago | 1 author, 1 change
public override float? CalculerDistance(IDimension dimensionVehiculeRef)
{
    DistanceDimension = base.CalculerDistance(dimensionVehiculeRef);

    if (DistanceDimension == 1)
    {
        DistanceDimension = PercentDiffDimension(dimensionVehiculeRef);
    }

    return DistanceDimension;
}
```

Dans la classe DimensionInteger, nous calculons chaque distance entre deux entiers (comme le nombre de places). Dans le cas où l'égalité parfaite n'est pas concluante.

Cela s'applique à toutes les dimensions qui ont des entiers comme valeurs. Le même principe est utilisé pour les nombres flottants.

```
/// <summary>
/// Calcul le pourcentage de différence entre les deux nombres. Nécessairement entre 0 et 1 représentant la distance
/// </summary>
/// <param name="dimensionVehiculeRef"></param>
/// <returns>Float entre 0 et 1</returns>
1 reference | Cedric Caudron, 222 days ago | 1 author, 2 changes
private float PercentDiffDimension(IDimension dimensionVehiculeRef)
{
    int diff = Math.Abs((int)dimensionVehiculeRef.Value - Value);
    if ((int)dimensionVehiculeRef.Value > Value)
    {
        var result = (float)diff / System.Convert.ToSingle(dimensionVehiculeRef.Value);
        return (float)result;
    }
    else
    {
        var result = (float)diff / (float)Value;
        return (float)result;
    }
}
```

10.4: Distance dimension finition

```
19 references | Caudron, 188 days ago | 2 authors, 2 changes
public override float? CalculerDistance(IDimension dimensionVehiculeRef)
{
    DistanceDimension = base.CalculerDistance(dimensionVehiculeRef);

    if (DistanceDimension == 1)
    {
        DistanceDimension = Utilitaires.Strings.Levenshtein.GetLevenshteinIndice(Value, (string)dimensionVehiculeRef.Value);
    }

    return DistanceDimension;
}
```

Pour la finition, on utilise l'indice basé sur la distance de Levenshtein.

Calcul de l'indice de Levenshtein grâce au package Nuget Fastenshtein.

```
2 references | Caudron, 203 days ago | 2 authors, 2 changes
public static class Levenshtein
{
    2 references | Caudron, 203 days ago | 2 authors, 2 changes
    public static float GetLevenshteinIndice(string value1, string value2)
    {
        int levenshteinDistance = Fastenshtein.Levenshtein.Distance(value1, value2);

        return (float)levenshteinDistance / Math.Max((float)value2.Length, (float)value1.Length);
    }
}
```

10.5: Distance dimension libellé

```
19 references | Caudron, 90 days ago | 2 authors, 4 changes
public override float? CalculerDistance(IDimension dimensionVehiculeRef)
{
    DistanceDimension = base.CalculerDistance(dimensionVehiculeRef);

    if (DistanceDimension == 1)
    {
        var distanceLibelle = new CalculDistanceLibelle(GetValueInTopDimension(), (string)dimensionVehiculeRef.Value);

        distanceLibelle.Execute();

        DistanceDimensionString = 1 - distanceLibelle.GetDistance();

        DistanceDimensionStringLevenshtein = Utilitaires.Strings.Levenshtein.GetLevenshteinIndice(GetValueInTopDimension(), (string)dimensionVehiculeRef.Value);

        var ponderationStr = PonderationsConfig.PonderationDistanceString;
        var ponderationLev = PonderationsConfig.PonderationDistanceStringLevenshtein;

        DistanceDimension =
            (ponderationStr * DistanceDimensionString + ponderationLev * DistanceDimensionStringLevenshtein) / (ponderationLev + ponderationStr);
    }

    return DistanceDimension;
}
```

Et pour terminer avec les exemples de calcul de distance, voici celui pour le libellé, qui, une fois de plus, est le plus complexe car il a, je le rappelle, deux distances différentes, et sa distance générale est la moyenne pondérée de ces deux distances.

Ci-dessus, le processus de calcul de la distance générale, en récupérant les deux sous-distances ainsi que les pondérations.

```
1 reference | Cedric Caudron, 287 days ago | 1 author, 1 change
internal void Execute()
{
    CompareStrings compareVehicule = new CompareStrings();
    CompareStrings compareVehiculeReferentiel = new CompareStrings();

    _listWordVehicule = compareVehicule.CutString(_libelleVehicule);
    _listWordReferentiel = compareVehiculeReferentiel.CutString(_libelleVehReferentiel);

    _sumDistances = compareVehicule.GetSumDistances(_listWordVehicule, _listWordReferentiel);
}

1 reference | Cedric Caudron, 284 days ago | 1 author, 2 changes
internal float? GetDistance()
{
    int divide;

    if ((_listWordReferentiel.Count != 0) && (_listWordVehicule.Count != 0))
    {
        if (_listWordReferentiel.Count >= _listWordVehicule.Count)
        {
            divide = _listWordReferentiel.Count;
        }
        else
        {
            divide = _listWordVehicule.Count;
        }
        return _sumDistances / divide;
    }
    else
    {
        return 0F;
    }
}
```

Et la classe CalculDistanceLibelle qui va calculer la distance entre les libellés « mot par mot ». Donc un ratio entre le nombre de mots communs entre les deux libellés à comparer sur le nombre de mot le plus important entre les deux libellés.

10.6: Calcul de la distance générale

```
/// <summary>
/// Mets à jour la distance générale du véhicule du référentiel par rapport au véhicule à tester
/// </summary>
/// <param name="dim"></param>
4 references | Cedric Caudron, 260 days ago | 1 author, 2 changes
public void CalculDistanceGenerale(IDimension dim)
{
    Distance += (float)dim.Ponderation * (float)dim.DistanceDimension;
}
```

Et enfin, nous pouvons mettre à jour la distance générale portée par la Liste des dimensions du véhicule du référentiel en ajoutant la valeur pondérée de la distance que l'on a calculé.

```
23 references | Caudron, 130 days ago | 3 authors, 10 changes
public static class PonderationsConfig
{
    public static float Marque = 1;

    public static float Modele = 1;

    public static float Energie = 1;

    public static float BoiteVitesse = 1;

    public static float Carosserie = 0;

    public static float Version = 1;

    public static float LibelleAnnonce = 1;

    public static float PlaqueImmatriculation = 1;

    public static float NombrePlaces = 1;

    public static float NombrePortes = 1;

    public static float DateMEC = 1;

    public static float PuissanceFiscale = 2;

    public static float PuissanceDin = 3;

    public static float CO2 = 0F;

    public static float Cm3 = 1;

    public static float PrixTTC = 1;

    public static float PrixHt = 1;

    public static float Finition = 1;

    public static float Generation = 1;

    public static float Phase = 1;

    public static float PonderationDistanceString = 1;

    public static float PonderationDistanceStringLevenshtein = 1;

    public static float Poids = 1;

    public static float Consommation = 1F;
}
```

Ci-contre la liste des pondérations des différentes dimensions, montrant celles pour lesquelles nous avons voulu donner plus de poids (déterminé en fonction des résultats que nous avons observé par la suite).

8 : La sélection des résultats

11.1: Tri par distance

```
// Tri par distance ascendante
List<ListDimensionsVehicule> dimensionTriees =
                    dimensionsVehRef
                    .OrderBy(d => d.Distance)
                    .ToList();
```

Il nous reste enfin à trier et sélectionner les véhicules que nous voulons faire renvoyer par l'API. Ci-contre le tri des résultat par distance croissante.

11.2: Sélection limite

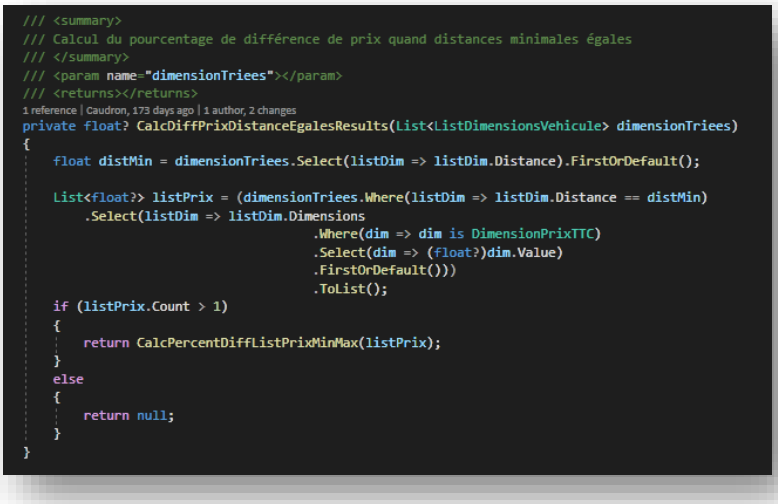
Ci-dessous, la méthode qui me permet de ne sélectionner que les véhicules qui ont une distance inférieure à un pourcentage (nous avons mis ce seuil à 10% pour le moment) de la distance minimale.

Ainsi, si le véhicule avec la distance minimale a une distance de 0.5, seuls seront renvoyés les véhicules qui auront une distance <= 0.55.

```
/// <summary>
/// Tri de la liste de dimension en fonction d'une valeur de delta à définir
/// </summary>
/// <param name="dimensionsVehiculeATester"></param>
/// <param name="dimensionTriees"></param>
/// <returns>Dimension triée et réduite en fonction de la distance définie</returns>
1 reference | Caudron, 172 days ago | 2 authors, 9 changes
private List<ListDimensionsVehicule> TriQueLesProbables(ListDimensionsVehicule dimensionsVehiculeATester, List<ListDimensionsVehicule> dimensionTriees)
{
    if (_vehicule.QueLesProbables)
    {
        float distanceTri = dimensionTriees[0].Distance + (BornesConfig.DeltaPercentSelectionDistance * dimensionTriees[0].Distance / 100);

        dimensionTriees = dimensionTriees
            .Where(d => d.Distance <= distanceTri)
            .ToList();

        if (dimensionsVehiculeATester.IdArgus != null && !VerifyIfIdArgusPresentInList(dimensionTriees, (int)dimensionsVehiculeATester.IdArgus))
        {
            _errorsList.Add("L'id Argus attendu a une distance > à " + BornesConfig.DeltaPercentSelectionDistance + " % par rapport au premier résultat", ErrorType.Tri);
        }
    }

    return dimensionTriees;
}
```

On peut noter également la deuxième partie de la méthode, qui fait partie du débogage, et dont j'ai dit que je ne parlerais pas ou succinctement. Dans ce cas, si on a un id argus à comparer en entrée, il vérifie s'il est présent dans la liste des résultats, et si ce n'est pas le cas, rajoute une erreur. Il existe une multitude de traitement de ce genre pour vérifier la bonne marche de l'algorithme, et à différentes étapes de son exécution.

11.3: Pourcentage de différence de prix

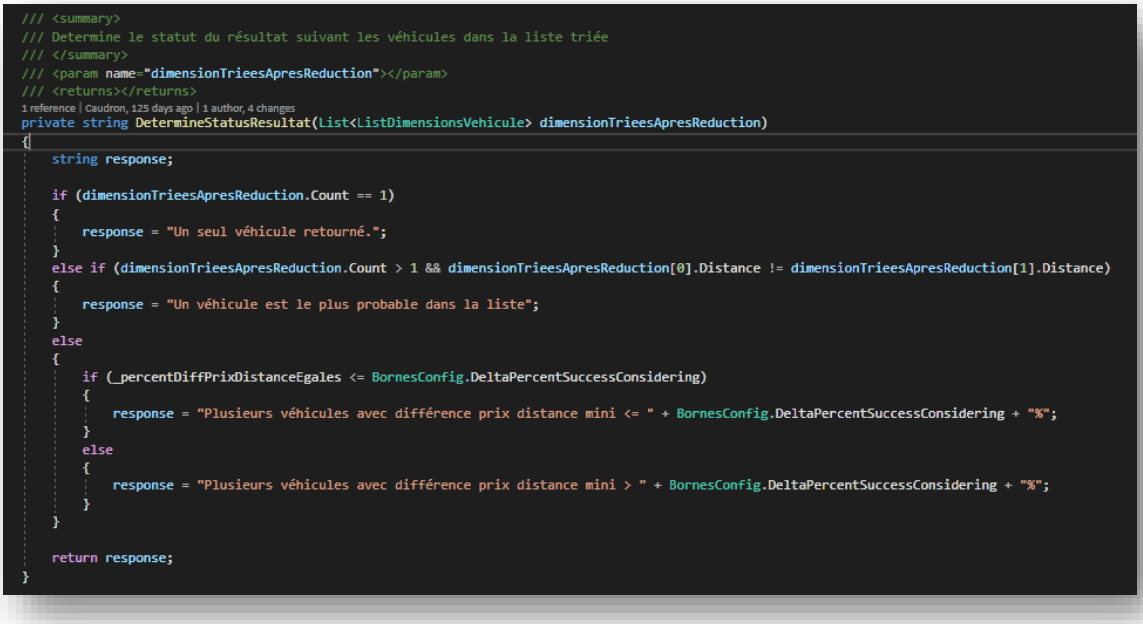


```
/// <summary>
/// Calcul du pourcentage de différence de prix quand distances minimales égales
/// </summary>
/// <param name="dimensionTriees"></param>
/// <returns></returns>
1 reference | Caudron, 173 days ago | 1 author, 2 changes
private float? CalcDiffPrixDistanceEgalesResults(List<ListDimensionsVehicule> dimensionTriees)
{
    float distMin = dimensionTriees.Select(listDim => listDim.Distance).FirstOrDefault();

    List<float?> listPrix = (dimensionTriees.Where(listDim => listDim.Distance == distMin)
        .Select(listDim => listDim.Dimensions
                                  .Where(dim => dim is DimensionPrixTTC)
                                  .Select(dim => (float?)dim.Value)
                                  .FirstOrDefault()))
                                  .ToList();
    if (listPrix.Count > 1)
    {
        return CalcPercentDiffListPrixMinMax(listPrix);
    }
    else
    {
        return null;
    }
}
```

De plus, pour chaque liste de véhicules sélectionnés pour être renvoyé, on calcule le pourcentage de différence de prix entre le plus cher et le moins cher, ainsi que le pourcentage de différence de prix entre les véhicules qui, le cas échéant, ont une distance minimale égale. C'est un indicateur, pour le client (ce n'est pas l'algorithme qui prend la responsabilité de trier par rapport à lui), de la différence entre les véhicules sélectionnés et ainsi de décider si oui ou non il utilisera cette réponse pour ses futurs traitements.

9 : Le formatage de la réponse

12.1: Les différents statuts de réponse



```
/// <summary>
/// Determine le statut du résultat suivant les véhicules dans la liste triée
/// </summary>
/// <param name="dimensionTrieesApresReduction"></param>
/// <returns></returns>
1 reference | Caudron, 125 days ago | 1 author, 4 changes
private string DetermineStatusResultat(List<ListDimensionsVehicule> dimensionTrieesApresReduction)
{
    string response;

    if (dimensionTrieesApresReduction.Count == 1)
    {
        response = "Un seul véhicule retourné.";
    }
    else if (dimensionTrieesApresReduction.Count > 1 && dimensionTrieesApresReduction[0].Distance != dimensionTrieesApresReduction[1].Distance)
    {
        response = "Un véhicule est le plus probable dans la liste";
    }
    else
    {
        if (_percentDiffPrixDistanceEgales <= BornesConfig.DeltaPercentSuccessConsidering)
        {
            response = "Plusieurs véhicules avec différence prix distance mini <= " + BornesConfig.DeltaPercentSuccessConsidering + "%";
        }
        else
        {
            response = "Plusieurs véhicules avec différence prix distance mini > " + BornesConfig.DeltaPercentSuccessConsidering + "%";
        }
    }

    return response;
}
```

Ainsi on peut considérer le succès si un seul véhicule est retourné, si un véhicule a une distance minimale sans égalité, et donner l'indication par rapport à la différence de prix et lui fixer un seuil pour considérer que c'est un succès ou non.

12.2: La réponse

```
/// <summary>
/// Formatte la réponse de l'algo que renverra l'API
/// </summary>
/// <param name="dimensionTrieesApresReduction"></param>
/// <returns>CalculIdArgusDTO</returns>
1 reference | Caudron, 95 days ago | 2 authors, 12 changes
private CalculIdArgus FormatResponse(List<ListDimensionsVehicule> dimensionTrieesApresReduction, ListDimensionsVehicule dimensionsVehATester)
{
    CalculIdArgus response = new CalculIdArgus
    {
        NbreResultat = dimensionTrieesApresReduction.Count,

        DeltaDiffPrixAllResults = _percentDiffPrix,

        DeltaPrixDistancesEgales = _percentDiffPrixDistanceEgales,

        Resultats = new List<ResultatIdArgus>(),

        DebugLibelleEnvoyeNettoye = dimensionsVehATester.Dimensions
                            .Where(dim => dim is DimensionLibelleAnnonce)
                            .Select(dim => (string)dim.Value)
                            .FirstOrDefault()
    };

    if (dimensionTrieesApresReduction.Count != 0)
    {
        response.Succes = true;

        response.IdFound = DetermineStatusResultat(dimensionTrieesApresReduction);

        response.Errors = _errorsList;

        response.NbMissingImportantEntries = (int)dimensionsVehATester.NbreMissingImportantEntries;

        int i = 0;

        foreach (ListDimensionsVehicule listDims in dimensionTrieesApresReduction)
        {
            response.Resultats.Add(new ResultatIdArgus
            {
                IdArgus = (int)listDims.IdArgus,
                Distance = listDims.Distance,
                Marque = listDims.Vehicule.Marque,
                Modele = listDims.Vehicule.Modele,
                SousModele = listDims.Vehicule.SousModele,
                Generation = listDims.Vehicule.Generation,
                Phase = listDims.Vehicule.Phase,
                Finition = listDims.Vehicule.Finition,
                DateMEC = listDims.Vehicule.DateMiseCirculation,
                DateDebutPeriode = listDims.Vehicule.DateDebutPeriode,
                DateFinPeriode = listDims.Vehicule.DateFinPeriode,
                DateDebut = listDims.Vehicule.DateDebut,
                DateFin = listDims.Vehicule.DateFin,
                Energie = listDims.Vehicule.Energie,
```

```
                Energie = listDims.Vehicule.Energie,
                BoiteVitesse = listDims.Vehicule.BoiteVitesse,
                TypeBoiteVitesse = listDims.Vehicule.TypeBoiteVitesse,
                Carrosserie = listDims.Vehicule.Carrosserie,
                Libelle = listDims.Vehicule.LibelleAnnonce,
                NombrePortes = listDims.Vehicule.NbPortes,
                NombrePlaces = listDims.Vehicule.NbPlaces,
                PuissanceDIN = listDims.Vehicule.PuissanceDin,
                PuissanceFiscale = listDims.Vehicule.PuissanceFiscale,
                PrixHt = listDims.Vehicule.PrixHt,
                PrixTTC = listDims.Vehicule.PrixTTC,
                Poids = listDims.Vehicule.Poids,
                Consommation = listDims.Vehicule.Consommation,
                CO2 = listDims.Vehicule.Co2,
                Cm3 = listDims.Vehicule.Cm3,
                Debug = new Debug
                {
                    DistanceBoiteVitesse = listDims.Dimensions
                            .Where(dim => dim is DimensionBoiteVitesse)
                            .Select(dim => dim.DistanceDimension)
                            .FirstOrDefault(),

                    DistanceCarrosserie = listDims.Dimensions
                            .Where(dim => dim is DimensionCarrosserie)
                            .Select(dim => dim.DistanceDimension)
                            .FirstOrDefault(),

                    DistanceCm3 = listDims.Dimensions
                            .Where(dim => dim is DimensionCm3)
                            .Select(dim => dim.DistanceDimension)
                            .FirstOrDefault(),

                    DistanceCo2 = listDims.Dimensions
                            .Where(dim => dim is DimensionCO2)
                            .Select(dim => dim.DistanceDimension)
                            .FirstOrDefault(),

                    DistanceEnergie = listDims.Dimensions
                            .Where(dim => dim is DimensionEnergie)
                            .Select(dim => dim.DistanceDimension)
                            .FirstOrDefault(),

                    DistanceLibelle = listDims.Dimensions
                            .Where(dim => dim is DimensionLibelleAnnonce)
                            .Select(dim => dim.DistanceDimension)
                            .FirstOrDefault(),

                    DistanceNbrePlaces = listDims.Dimensions
                            .Where(dim => dim is DimensionNombrePlaces)
                            .Select(dim => dim.DistanceDimension)
                            .FirstOrDefault(),
```

Ci-dessus et ci-contre une partie du formatage de la réponse, tout d'abord dans son entité, qui sera renvoyée au controller qui la mappera facilement en son équivalent DTO.

Dans la partie Debug, que je n'ai pas beaucoup abordé, j'y ajoute les différentes distances des dimensions, afin de pouvoir vérifier une à une s'il y a des soucis dans leur calcul.

Et enfin voici la représentation Swagger de la réponse renvoyée par l'API. Elle comprend toutes les informations des véhicules rapprochés ainsi que celles dont j'aurai besoin pour détecter d'éventuels problèmes.

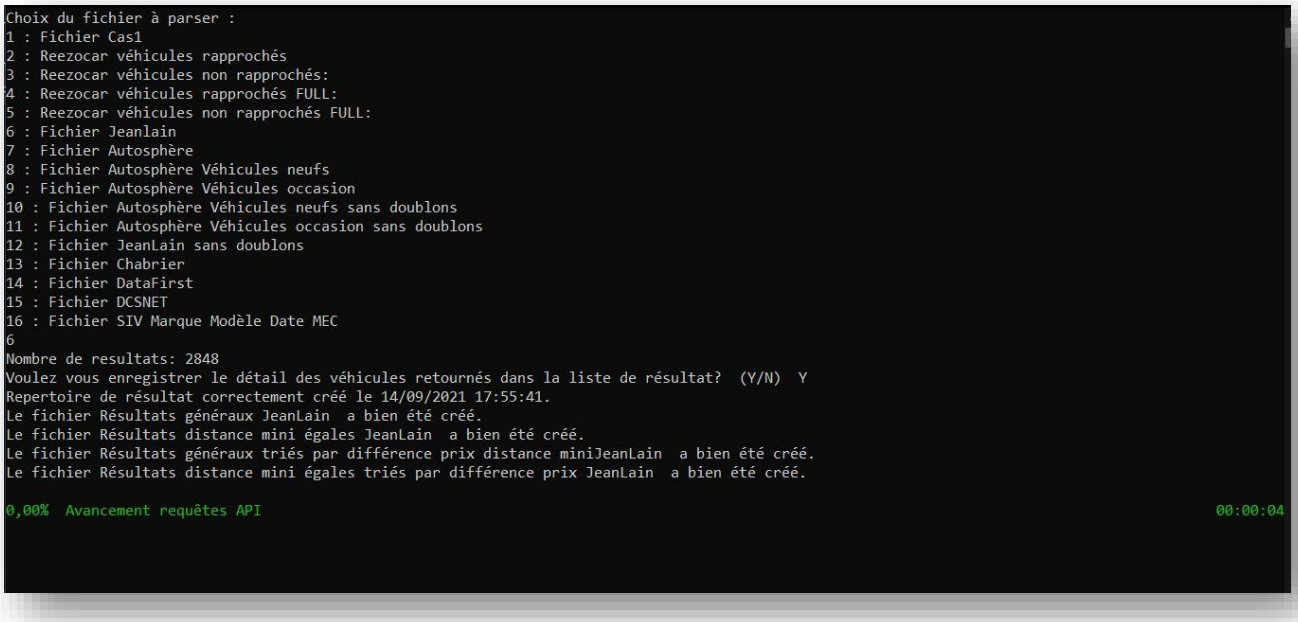
Ainsi se termine cette partie sur les processus de l'algorithme afin de trouver au mieux les véhicules et leur id Argus à partir des caractéristiques. Comme j'ai dit à plusieurs reprise, ce n'est pas totalement exhaustif car sinon j'aurais été obligé d'augmenter considérablement la taille de ce dossier, mais il donne un bon aperçu de l'ensemble des traitements qui ont été nécessaires pour arriver à ce résultat.

Néanmoins, je vais aborder succinctement dans la prochaine partie le client que j'ai développé afin d'interroger automatiquement l'API à partir de fichiers csv, et les fichiers résultats qu'il a généré afin de tester et améliorer l'algorithme au fil du temps.

V : LE CLIENT

1: Le principe

J'ai développé une application console pour tester l'API. Elle aura pour mission de parser les fichiers csv des partenaires (voir exemples page 10), de mettre en forme au minima les données pour qu'elles soient compatibles avec l'API, de lancer les requêtes vers l'API à proprement parler, de récupérer la réponse, de faire différentes statistiques, et enfin de formatter le tout et générer un fichier csv avec tous les résultats pour pouvoir visualiser en un coup d'œil les données en entrée, les données en sortie, ainsi que les informations de débogage.



Le client permet de choisir entre différents fichiers, qui n'ont pas les mêmes informations au mêmes endroits ou sous la même forme, et donc le choix déclenche la configuration spécifique du parsing ainsi que le parsing lui-même du fichier sélectionné.

Il effectue donc ensuite les requêtes véhicule par véhicule, traite les réponses et génère le fichier résultat qui comprend beaucoup d'étapes de traitements, les différentes distances etc.

2: Le fichier résultats

	A	B	C	D	E	F	G
1	No Vehicule uniqu	No Vehicul	Reponse algo Dsquad	Statut Id Argus envoye	Succes Diff Prix selon Algo DSQUA	Id Argus envoye	Reconciliation DSQUAD/Autosphere distance mini uniqu
2	1	1	Un seul véhicule retourné.	Premier résultat sans distance égale	True	True	True
3	2	2	Un véhicule est le plus probable dans la liste	Premier résultat sans distance égale	False	True	True
4		2					
5		2					
6	3	3	Un seul véhicule retourné.	Premier résultat sans distance égale	True	True	True
7	4	4	Plusieurs véhicules avec différence prix distance mini > 5%	Premier résultat avec distance égale	False	True	False
8		4					

Reconciliation DSQUAD/Autosphere distance mini egale	Id argus Autospher	Ids Argus DSQUA	Nbre resultats DSQUA	Distance mini DSQUA	Nb distances ajoutee	Marque envoye	Marque retourne	Modele envoy	Modele retourn	Sous modele retourn
	2029795	2029795	1	0,67156863	0	BMW	BMW	X7	X7	X7
True	1982747	1982747	3	0,41651207	0	AUDI	AUDI	Q2	Q2	Q2
		1606447	3	0,43913043	0		AUDI		Q2	Q2
		2011289	3	0,45555556	0		AUDI		Q2	Q2
	2126178	2126178	1	0,5079365	0	AUDI	AUDI	A1 Sportback	A1	A1 Sportback
True	1715918	1715918	2	1,1868131	0	VOLKSWAGEN	VOLKSWAGEN	Tiguan	Tiguan	Tiguan
		1715914	2	1,1868131	0		VOLKSWAGEN		Tiguan	Tiguan

Boite vitesse envoye	Boite vitesse retourne	Distance boite vitess	Type boite vitess	Libelle envoye	Libelle retourne	Dist generale libell	Dist levenshtein libell
Automatique	BVA8	0	Automatique	X7 40iA xDrive 340ch Exclusive	Génération I Phase Ph1 40iA xDrive 340ch Exclusive	0,67156863	0,6764706
Rob double embray	BVRD7	0	Automatique	Q2 35 1.4 TFSI 150ch COD S line S tronic 7	Génération I Phase Ph1 35 1.4 TFSI 150ch COD S line S tronic 7	0,41651207	0,46938777
	BVRD7	0	Automatique		Génération I Phase Ph1 1.4 TFSI 150ch COD S line S tronic 7	0,43913043	0,47826087
	BVRD7	0	Automatique		Génération I Phase Ph1 35 TFSI 150ch COD S line S tronic 7	0,45555556	0,51111114
Rob double embray	BVRD7	0	Automatique	A1 Sportback 30 TFSI 116ch Advanced S tronic 7	Génération II Phase Ph1 30 TFSI 116ch Advanced S tronic 7	0,5079365	0,5714286
Manuelle	BVM6	0	Manuelle	Tiguan 2.0 TDI 150ch Carat Exclusive 4Motion	Génération II Phase Ph1 2.0 TDI 150ch Carat Exclusive 4Motion	0,59340656	0,61538464
	BVM6	0	Manuelle		Génération II Phase Ph1 2.0 TDI 150ch Carat 4Motion	0,59340656	0,61538464

Dist compare libell	Libelle envoye nettoye	Libelle retourne nettoye	Finition envoye	Finition retourne	Dist Finitio	Date MEC	Date debut	Date fin
0,6666666	40IA XDRIVE	GENERATION I PHASE PH1 40IA XDRIVE		Exclusive	0	27/12/2019 00:00	17/10/2018 00:00	
0,36363637	35 1.4 TFSI COD S TRONIC 7	GENERATION I PHASE PH1 35 1.4 TFSI COD S TRONIC 7		S line	0	26/06/2019 00:00	28/06/2018 00:00	13/09/2018 00:00
0,39999998		GENERATION I PHASE PH1 1.4 TFSI COD S TRONIC 7		S line	0		28/07/2016 00:00	28/06/2018 00:00
0,39999998		GENERATION I PHASE PH1 35 TFSI COD S TRONIC 7		S line	0		13/09/2018 00:00	08/11/2018 00:00
0,44444442	30 TFSI S TRONIC 7	GENERATION II PHASE PH1 30 TFSI S TRONIC 7		Advanced	0	12/02/2020 00:00	14/06/2019 00:00	03/09/2020 00:00
0,57142854	2.0 TDI 4MOTION	GENERATION II PHASE PH1 2.0 TDI 4MOTION		Carat Exclusive	0	30/06/2017 00:00	27/04/2017 00:00	13/04/2018 00:00
0,57142854		GENERATION II PHASE PH1 2.0 TDI 4MOTION		Carat	0		27/04/2017 00:00	13/04/2018 00:00

Nbre jours entre date MEC et fin period	Energie envoye	Energie retourne	Dist Energi	Carrosserie envoye	Carrosserie retourne	Dist Carrosseri	Nbre Portes envoye	Nbre Portes retourne	Dist Nbre Porte	Nbre Places envoye	Nbre places retourne
	Essence	Essence	0	Break	SUV	1	5	5	0	7	7
	Essence	Essence	0	Break	SUV	1	5	5	0	5	5
		Essence	0		SUV	1		5	0		5
		Essence	0		SUV	1		5	0		5
	Essence	Essence	0	Berline	Berline	0	5	5	0	5	5
	Diesel	Diesel	0	S.U.V.	SUV	0	5	5	0	5	5
		Diesel	0		SUV	0		5	0		5

Dist Nbre Place	Puissance DIN envoye	Puissance DIN retourne	Dist Puissance DI	Puissance Fiscale envoye	Puissance Fiscale retourne	Dist Puissance Fiscal	CO2 Envoy	CO2 Retourn	Dist CO2	CO2 vÃ©hicule ajoute Partenaire meilleur	Poids Envoy
0		340	0		23		198	198	0		
0		150	0		8		123	123	0		
0		150	0		8			123	0		
0		150	0		8			123	0		
0		116	0		6		106	106	0		
0		150	0		8		141	141	0		
0		150	0		8			141	0		

Poids Retourn	Dist Poid	Consommation Envoye	Consommation Retourne	Dist Consommatio	Pays	Type Vendeu	Prix Vehicule Envoy	Prix Retourn	Difference Prix avec retour Autospher	Difference Prix min/max tout resulta	Difference Prix min/max des dis
3155			8,7				100750	100750	0%		
1840			5,4				36810	36810	0%	4,43%	
1840			5,4					35250			
1840			5,4					36810			
1680			4,9				27480	27260	0,80%		
2180			5,4				43870	43870	0%	6,04%	6,04
2180			5,4					41370			

Resultat Autospher	Nbre Erreu	Erreurs	Differences infos envoye vs infos referentiel id argus partenair
	0		0
	0		0
	0		
	0		
	0		1
	0		0
	0		

Je n'ai représenté ici que les résultats de 4 véhicules. Le fichier étant très long, et les fichiers analysés pouvant avoir plusieurs dizaines de milliers de véhicule, je ne peux bien évidemment pas en mettre un en entier.

Mais ces exemples sont représentatifs de trois cas en particulier. Celui où l'algorithme ne renvoie qu'un véhicule, celui où il y a plusieurs résultats mais le premier véhicule a une distance minimum, et enfin le cas où il y a plusieurs véhicules avec distance minimale égale, ce qui est le cas sur lequel on ne peut pas trancher avec certitude quel est le bon véhicule, d'autant plus que la différence de prix de ces véhicules est > 5%, seuil qui a été fixé pour déterminer une certaine fiabilité du résultat.

3: Les analyses

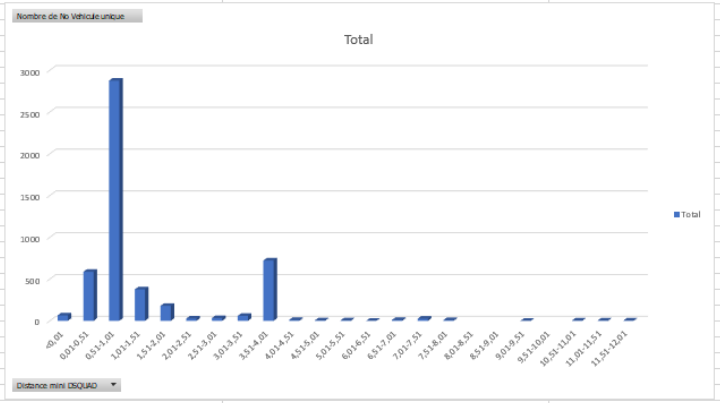
L'analyse de ce genre de tableau m'aura finalement pris la majorité de mon temps. Enormément d'analyse, de statistiques diverses, afin d'améliorer l'algorithme au fur et à mesure. J'estime que mon temps a été partagé entre environ 90% d'analyse, et 10% de conception et écriture du code.

Grace aux tableaux croisés dynamiques d'Excel (dont il y a un exemple page suivante), j'ai pu recouper différentes données et, avec des réunions régulières, nous avons pu avancer pour arriver à un résultat qui est pour ma part assez satisfaisant. On peut toujours améliorer l'algorithme, mais nous serons toujours confrontés au plafond de verre de la qualité de donnée. En effet, il y a régulièrement des erreurs dans les fichiers envoyés par les partenaires… Erreurs impossibles à détecter de manière automatique, et qui fausse du coup le résultat. Mais dans l'ensemble, mon algorithme effectue le travail, le fait bien, et il obtient dans beaucoup de cas de meilleurs résultats que les solutions en place jusqu'à présent.

Étiquettes de lignes	Nombre de No Vehicule unique	Pourcentage
A une distance > au pourcentage de la distance minimale	82	1,76%
Dans la liste mais n'a pas la distance minimale	266	5,15%
Distance mini OK mais pas ce véhicule renvoyé par partenaire	663	13,01%
Id Argus envoyé non trouvé. Pb date MEC?	180	3,72%
Le partenaire a trouvé un véhicule mais pas DSQUAD	65	1,16%
Premier résultat avec distance égale	954	19,52%
Premier résultat sans distance égale	2790	55,68%
Total général	**5000**	**100,00%**

Croisement statut DSQUAD avec retour Cas1	Étiquettes de colonnes					
Étiquettes de lignes	**A une distance > au pourcentage de la distanc**	**Dans la liste mais n'a pas la distance**	**Distance mini OK mais pas ce véhicule renvoyé par**	**Id Argus envoyé non trouvé. Pb date**	**Le partenaire a trouvé un véhicule mais pas**	**Premier résultat avec distance é**
Aucun véhicule trouvé					65	
Plusieurs véhicules avec différence prix distance mini <= 5%	12	44	296	45		522
Plusieurs véhicules avec différence prix distance mini > 5%	15	37	367	41		432
Un seul véhicule retourné.	50			47		
Un véhicule est le plus probable dans la liste	5	185		47		
Total général	**82**	**266**	**663**	**180**	**65**	**954**

Répartition des distances	Nombre de No Vehicule unique
<0,01	65
0,01–0,51	587
0,51–1,01	2875
1,01–1,51	376
1,51–2,01	179
2,01–2,51	27
2,51–3,01	33
3,01–3,51	59
3,51–4,01	722
4,01–4,51	10
4,51–5,01	5
5,01–5,51	5
6,01–6,51	1
6,51–7,01	10
7,01–7,51	26
7,51–8,01	9
8,01–8,51	
8,51–9,01	
9,01–9,51	1
9,51–10,01	
10,51–11,01	3
11,01–11,51	4
11,51–12,01	3
Total général	**5000**

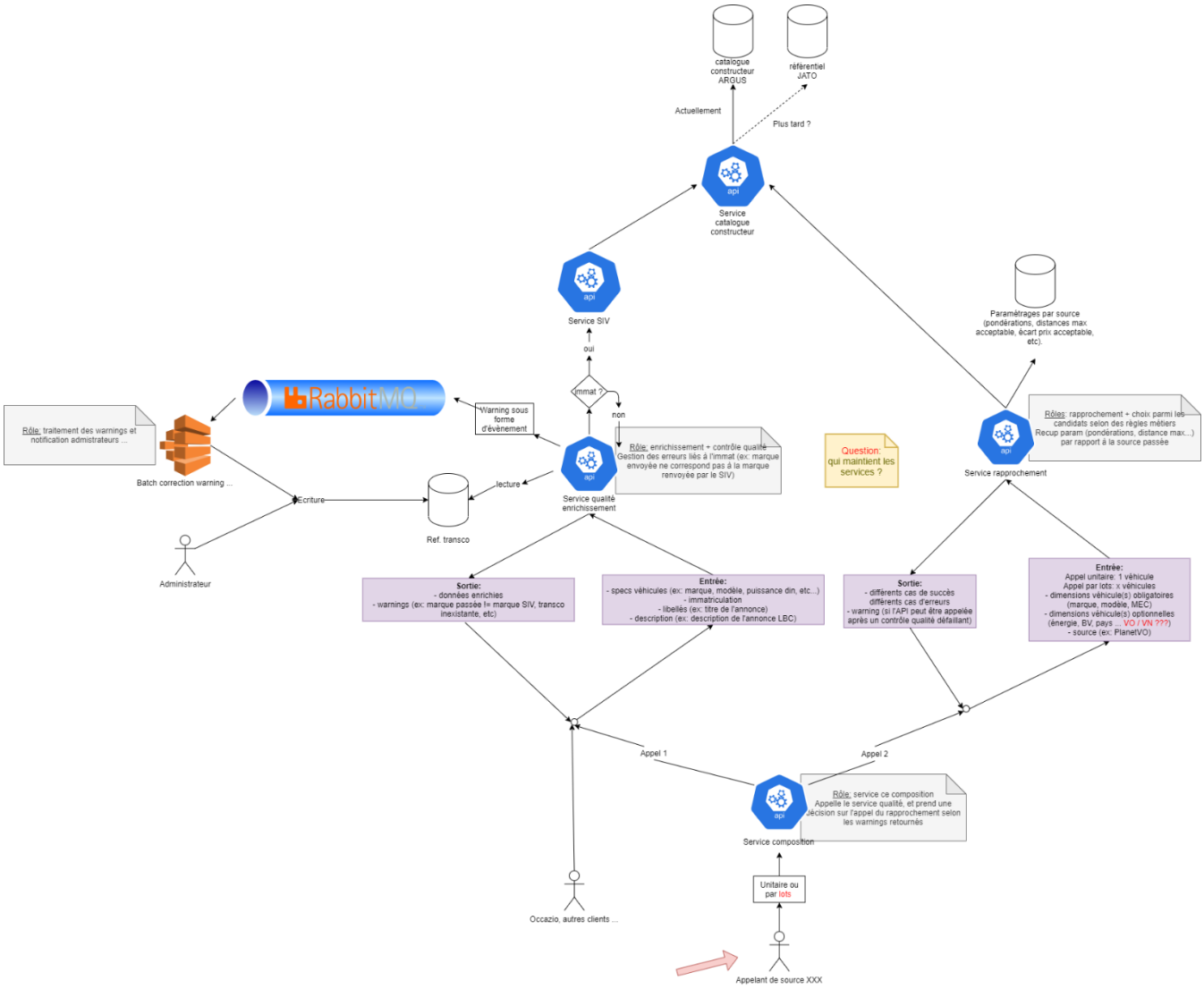


VI : LE FUTUR

1: Vers une scission ?

Il a été question de scinder en deux le rapprochement, afin de séparer la partie « qualité de donnée » et la partie rapprochement pur. De plus il devait être intégré dans un ensemble bien plus grand dont voici le schéma (crédit schéma : Benjamin Schmidt de la DSQUAD).

En se servant du service SIV (Système Immatriculation des Véhicules du ministère), on pensait pouvoir enrichir les données pour un meilleur rapprochement. Mais la qualité de cette base de données (sur laquelle je travaille depuis peu) n'est pas si bonne qu'espérée.



2: Mise à disposition en package Nuget ?

Pour le moment, ce serait la solution qui serait retenue. Transformer l'algorithme afin de le mettre à disposition en tant que package Nuget pour pouvoir l'intégrer dans différents outils des différents services de CGI Finance.

3: Intégration dans Occazio ?

La question est venue régulièrement pour une intégration de l'API dans l'application Occazio en remplacement du précédent algorithme en Python. L'API est prête logiquement pour le déploiement. Mais de nombreux problèmes de correction de bugs, couplé à une deadline rapprochée pour sortir une version en production l'ont mis en suspens pour l'instant. Mais il devrait y être intégré dès que tout se sera calmé.

VII : CONCLUSION

Ainsi nous arrivons au chapitre final de ce dossier. Conclusion d'un an de travail sur cet algorithme.

J'aurai appris énormément de chose sur cette année, dans la façon de découper un projet en plusieurs petits, dans la façon de réfléchir et d'écrire mon code. J'ai encore des progrès à faire pour atteindre plus de généricité, mais je suis sur la bonne voie et je compte bien continuer dans ce sens, en appliquant toujours plus les bonnes pratiques de développement.

J'aurai également légèrement amélioré mes compétences en réseau, avec l'utilisation de Linux, de machines virtuelles, de container Docker etc. C'est un domaine où je suis encore novice mais qui m'intéresse au plus haut point afin d'acquérir un panel de compétences le plus complet possible.

J'aurai découvert le travail en équipe (même si j'ai beaucoup été seul sur ce projet en particulier) et les principes d'un développement de type AGILE.

Et je tiens à remercier tout ceux qui m'auront fait confiance cette année, en particulier Laurent Boudet, Jawad Chahed, Benjamin Schmidt du coté DSQUAD , et tous les formateurs du coté Valarep.

Et je suis heureux de pouvoir continuer l'aventure au sein de la DSQUAD pour les deux prochaines années, toujours en alternance.