le cnam

CGI FINANCE

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL HAUTS DE FRANCE / VALENCIENNES

MEMOIRE

présenté en vue d'obtenir

le DIPLOME D'INGENIEUR CNAM

SPECIALITE : Informatique

OPTION : Systèmes d'Information et Business Intelligence

par

CAUDRON Cédric

Soutenu le 09/02/2024

Conception, développement, déploiement et optimisation d'un système pour la réconciliation et la mise en qualité des données de véhicules

---

JURY

PRESIDENTE :	**Mme Elisabeth METAIS**	**Professeur du CNAM**
MEMBRES :	**M. Didier NAKACHE**	**Coordinateur informatique**
	Mme Fanny FIEVET	**Responsable centre CNAM Valenciennes**
	M. Patrick COLIN	**Directeur régional CNAM HDF**
	M. Laurent BOUDET	**Chef du service DDI, CGI FINANCE**

Pour mon fils Corentin

Table des matières

Remerciements

Je souhaite adresser de très vifs remerciements aux personnes qui m'auront permis de mener à terme ce projet d'études au CNAM Valenciennes.

Je remercie les membres du jury, Elisabeth METAIS en sa qualité de présidente, Didier NAKACHE en tant que coordinateur informatique, Fanny FIEVET en sa qualité de responsable du CNAM de Valenciennes, ainsi que Patrick COLIN, en tant que directeur du CNAM Haut de France, pour leur implication dans cet examen, la lecture attentive de ce mémoire, et leur présence lors de la soutenance prochaine.

Mes remerciements au centre CNAM de Valenciennes pour leur disponibilité et leur implication pour répondre à nos différentes interrogations.

Un grand merci également à CGI Finance pour sa confiance réitérée pendant 3 ans, plus particulièrement à Laurent BOUDET, chef de service de la DDI, Benjamin SCHMITT, Tech Lead de la DSQUAD, ainsi qu'à Elyes SFAR et Marc HACHE, développeurs au sein de la DSQUAD.

D'un point de vue personnel, je souhaite bien évidemment remercier Mylaine ROULIN, ma compagne depuis un peu plus d'un an, qui m'aura supporté sur cette dernière année particulièrement difficile, elle aura été source de motivation et clairement, sans son soutien indéfectible, ainsi que celui de sa famille, je ne pense pas que j'aurai réussi à terminer ce parcours d'études ni ce mémoire !

Un grand merci également à mon fils Corentin, qui m'aura mis bien des bâtons dans les roues, me fait faire du sang d'encre, mais c'est aussi et surtout pour toi que j'ai repris mes études et que je me suis battu afin d'en être là aujourd'hui ! Et c'est donc pour cela que je te pardonne (presque) tout, et que je te dédies ce mémoire.

Et enfin, j'adresse un remerciement spécial à Mr HUMERY, psychologue à Douai, qui, bien avant ma reconversion, m'aura mis sur la voie d'une reprise d'études en informatique et sans qui rien n'aurait été possible…

1 INTRODUCTION

Dans un monde où la précision de l'évaluation des véhicules d'occasion est cruciale, le projet RMOD, mené au sein de CGI Finance, se positionne comme une initiative clé visant à améliorer la réconciliation et l'exactitude des données relatives aux véhicules d'occasion. Ce mémoire offre une exploration approfondie du projet, depuis ses fondements théoriques jusqu'à sa mise en œuvre concrète, en passant par les phases de développement, de test, de validation et d'exploitation.

Le projet débute par un état de l'art, qui examine les solutions existantes chez CGI Finance, mettant en évidence les besoins et les lacunes dans les pratiques actuelles et identifiant ainsi les opportunités d'innovation. Cette analyse sert de base pour la conception de RMOD, une réponse adaptée aux défis spécifiques du secteur.

La partie développement est essentielle dans ce mémoire, car elle détaille la transformation de la théorie en une application pratique robuste et fonctionnelle. Cette section met en lumière la méthodologie de développement adoptée, les choix technologiques, ainsi que les défis et solutions techniques rencontrés lors de la création des composants clés de RMOD.

Les phases de test et de validation soulignent l'importance accordée à la performance, la fiabilité et la conformité aux besoins spécifiques du secteur de la bancassurance. L'accent est mis sur les tests unitaires, d'intégration et de non-régression pour garantir une application sans faille.

L'exploitation de RMOD révèle la nécessité d'une surveillance continue et d'une analyse en temps réel pour une amélioration continue. Les outils et stratégies employés pour maintenir et optimiser la qualité du système sont explorés en détail.

Enfin, la section "Analyse et Validation" illustre comment les données récoltées et analysées, via une application d’automatisation, contribuent à l'ajustement et à l'amélioration de l'algorithme, soulignant l'importance de l'évaluation continue dans la réussite du projet.

Ce mémoire, en somme, offre un aperçu complet du projet RMOD, illustrant la manière dont CGI Finance a relevé les défis et transformé les opportunités en innovations concrètes et efficaces dans le domaine de la bancassurance.

2 PRESENTATION DE L'ENTREPRISE

2.1 Le groupe Société Générale

Figure 1 - Logo Société Générale

La Société Générale est une des principales et l'une des plus anciennes banques françaises. 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 ». [1]

Plus de 150 ans plus tard, elle est devenue un grand groupe financier international comportant plus de 25 millions de clients (particuliers, entreprises et investisseurs) et plus de 117 000 collaborateurs dans ses différentes filiales réparties dans 66 pays. [2]

C'est au sein de ce vaste ensemble que se trouve une filiale française spécialisée dans le financement automobile/bateau et regroupement de crédit : CGI Finance.

2.2 CGI Finance

2.2.1 Organisation générale

Figure 2 - Logo CGI Finance

Créé en 1951, CGI Finance est donc une filiale du groupe Société Générale spécialisée dans le financement.

Avec Ludovic VAN DE VOORDE comme actuel Directeur Général, elle compte actuellement plus de 1000 employés « financeurs créatifs », répartis entre le siège à Marcq en Baroeul et 19 agences commerciales en France, et couvrant plus de 160 métiers différents. [3]

Ses activités sont axées autour de 3 pôles métiers complémentaires :

- Le financement automobile (dans lequel nous avons évolué)
- Le financement nautique
- Le regroupement de crédit

L'entreprise se veut à la pointe de l'innovation technologique et digitale, en proposant divers outils à ses partenaires, développés et supportés par ses départements informatiques.

2.2.2 Organisation au sein de l'entreprise

Au sein de CGI Finance coexistent deux services informatiques distincts : la DOI et la Dsquad.

La première, la DOI, est principalement chargée des développements applicatifs de l'entreprise, se subdivisant en de multiples domaines selon leur spécialité. Certains services s'occupent d'applications spécifiques, tandis que d'autres jouent un rôle transversal pour assurer la cohérence du système d'information (S.I.).

Quant à la DSquad, dépendante de la Direction Data Intelligence, est celle où nous avons évolué durant ces quelques années.

La DDI est responsable de tout ce qui est traitement de données, ainsi que les aspects réglementaires de l'entreprise (respects des lois, provisions, etc…)

2.2.3 La Dsquad

Figure 3 - Logo de la DSquad

Au sein de la DDI, nous retrouvons un service distinct, la Dsquad, auquel nous appartenons. Composée d'une équipe fluctuant entre 5 et 10 membres selon les périodes, la Dsquad se distingue par son esprit "startup". En effet, c'est l'unique service au sein de CGI Finance qui bénéficie de la liberté de développer en dehors du SI, lui octroyant ainsi une autonomie notable. Cette latitude a pour objectif de s'affranchir de certaines contraintes, favorisant ainsi l'innovation.

Sous la direction de Laurent Boudet, directeur de la data intelligence, la Dsquad a piloté plusieurs projets novateurs. Les plus récents comprennent :

- Occazio : Application mobile Android permettant la mise en vente de véhicules d'occasion, ainsi que la mise en relation avec des concessionnaires partenaires. [4]
- RMOD : Ensemble d'outils pour la mise en qualité des données et de réconciliation avec un véhicule de la base Argus.

Nous approfondirons ces applications ensuite, étant donné, de plus, que RMOD constitue le cœur de ce mémoire.

2.3 Aperçu de l'applicatif existant

Dans cette partie, nous allons présenter succinctement quelques applications de CGI Finance, qui auront une incidence sur le projet RMOD, sujet de ce mémoire.

2.3.1 Occazio

Occazio, développée par la Dsquad, est une application mobile Android innovante qui souhaite faciliter la façon dont les véhicules d'occasion sont achetés et vendus, et ce en proposant une mise en relation directe sécurisée entre clients et acheteurs, qu'ils soient particuliers ou professionnels.

Elle intègre diverses fonctionnalités comme :

- la possibilité d'entrer ou de prendre en photo une plaque d'immatriculation pour identifier un véhicule et voir ses caractéristiques
- la possibilité d'inclure le rapport Histovec du véhicule pour sécuriser le potentiel acheteur
- la possibilité d'inspecter la carrosserie du véhicule grâce au partenaire Tchek afin de chiffrer d'éventuelles dépenses [5]
- la cotation automatique du véhicule, avec la comparaison à d'autres offres existantes afin de positionner le prix de vente comme compétitif ou non
- une messagerie intégrée permettant des échanges directs entre acheteurs et vendeurs

En conclusion, Occazio est le dernier outil de la stratégie numérique de CGI Finance à avoir été mis à disposition, et démontre bien les ambitions d'innovation de l'entreprise.

Figure 4 - Image promotionnelle d'Occazio

2.3.2 Téléfi

Téléfi 3, et Téléfi 4 son successeur, développés par la DOI, sont des outils d'aide à la vente mis à disposition dans les concessions partenaires.

Ces outils permettent de faire des simulations et des demandes de financements pour les véhicules sélectionnés.

Ainsi, à partir d'une plaque d'immatriculation, ils sont capables de retrouver le véhicule en question, de retrouver le prix dudit véhicule, et à partir de là de faire une simulation de la mensualité des différents types de financements possibles (Crédit classique, LOA, LDD etc…), et d'en faire directement la demande.

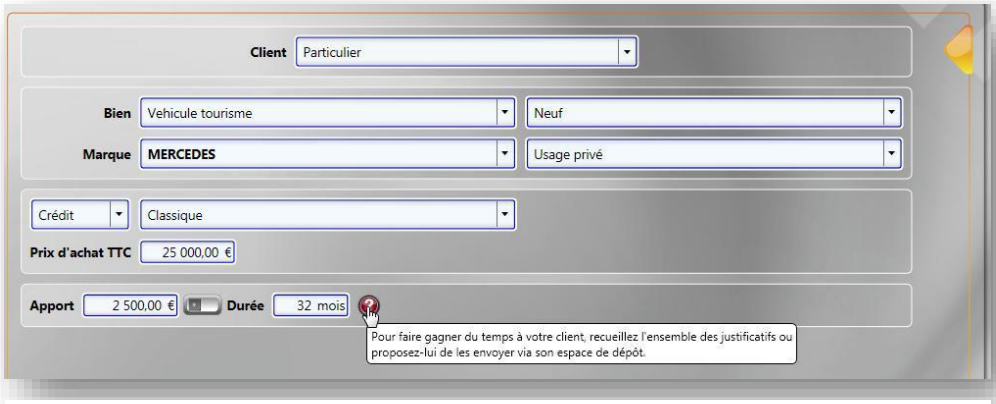


Figure 5 - Téléfi - Exemple d'écran de demande de financement

2.3.3 Vivafi & Vivacar

Vivafi est un outil de simulation de financement et d'acceptation en ligne sous la forme de modules intégrables dans des sites Web externes.

Il est ainsi utilisé par plus de 400 sites Web de partenaires, parmi eux Vivacar, qui est le site vitrine de CGI Finance intégrant des milliers de véhicules de partenaires, et bien évidemment l'outil Vivafi. [6]

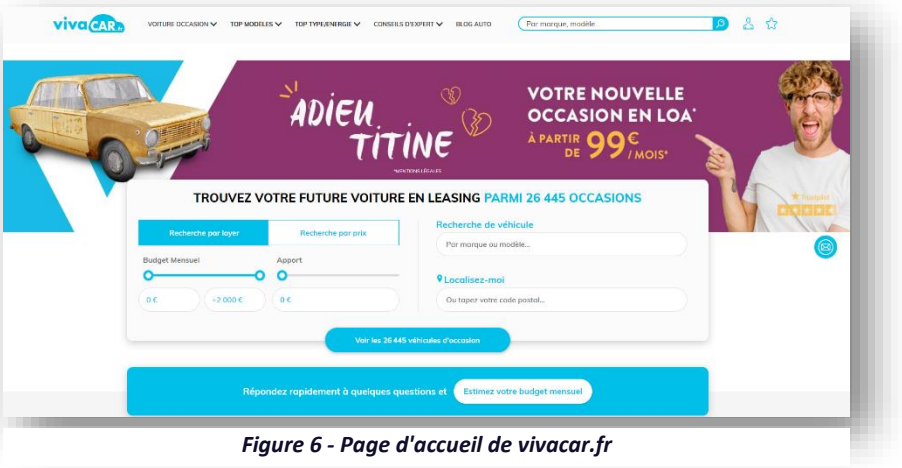


Figure 6 - Page d'accueil de vivacar.fr

3 ETAT DE L'ART

3.1 Besoins métier

3.1.1 Un financement au plus proche de la réalité

L'une des principales activités de CGI Finance concerne le financement de véhicules d'occasion, vendus par ses nombreux partenaires professionnels. Une des principales problématiques de ce secteur réside dans la détermination précise du prix du véhicule, compte tenu des informations variées et non standardisées fournies par ces partenaires.

Le besoin métier est donc d'estimer le prix initial du véhicule le plus fidèlement possible. À cette estimation, on applique une décote basée sur l'âge et le kilométrage du véhicule. C'est sur cette base recalculée que sont déterminées la mensualité et la VR (Valeur Résiduelle ou de Reprise). Il est crucial de noter que, généralement, une mensualité réduite conduit à une VR plus élevée. Par ailleurs, pendant la durée du financement, CGI demeure propriétaire du véhicule.

L'enjeu majeur est le suivant : si la VR est surestimée (ce qui conduit à des mensualités moindres), la revente ultérieure du véhicule peut poser problème. Si, à la fin du contrat, le client ne choisit pas de racheter le véhicule et si celui-ci est revendu à un prix inférieur à la VR, alors la société enregistre une perte.

Imaginons que l'estimation initiale ait été incorrecte, menant à une surévaluation du véhicule : cela pourrait dissuader le client à cause d'une mensualité élevée, et la VR fixée serait trop optimiste par rapport à la valeur réelle du marché. Une revente au tarif de la VR deviendrait alors complexe, voire irréalisable, et pourrait aboutir à une perte nette pour CGI.

Il est donc impératif d'évaluer avec précision et fiabilité le prix initial du véhicule. C'est dans ce contexte que l'élaboration d'un outil de réconciliation fiable s'avère essentielle.

Au sein de CGI, il y a le besoin de deux types de réconciliations distincts :

- La réconciliation par caractéristiques
- La réconciliation par immatriculation

3.1.2 La réconciliation par caractéristiques

Lorsque CGI Finance traite les informations des véhicules en provenance des systèmes de gestion de concessionnaires (DMS) de ses partenaires, elle est souvent confrontée à des données de qualité variable. Cette variabilité peut se traduire par une complétude insuffisante, une incohérence, ou une absence d'informations clés. De surcroît, ces données ne comportent généralement pas les identifiants ou références permettant une correspondance directe avec le référentiel Argus, sur lequel le SI de CGI Finance s'appuie.

Dans ce contexte, une réconciliation basée sur les caractéristiques des véhicules est cruciale. Elle permet d'établir des correspondances pertinentes entre les véhicules rapportés par les DMS et ceux du référentiel Argus, assurant ainsi une meilleure fiabilité dans les traitements ultérieurs. Cette démarche est d'autant plus essentielle que la qualité et la précision des données influencent directement les décisions financières.

modele	codeMarq	datePremiereN	kilome	puissanceFiscale	co2	Portes	Finition	Automatique	Gamme	immatriculation	Version	Nb_rapports	codeEnergie
CHIRON	BUG	01/06/2020	500	263.0	516.0g/km	nan	NOT_FOUND	VRAI	CHIRON	REUDBF4953	nan	nan	ESS
CRAFTER	VOK	01/05/2023	10	7.0	226.0g/km	nan	CLIM	FAUX	CRAFTER	REUB6A8C9F	nan	nan	GSL
IX3	BMW	01/03/2021	21500	18.0	nang/km	5.0	IMPRESSIVE	VRAI	IX3	REU9B0EAD8	BEV 80KWH IMPRESSIVE AUTO	nan	ELE
MASTER	REN	01/04/2021	28167	8.0	225.0g/km	nan	CONFORT	FAUX	MASTER	REU00853F0	nan	nan	GSL
DEFENDER	LAN	01/03/2022	20000	43.0	nang/km	5.0	V8 CARPATHIAN	VRAI	DEFENDER	REUC40B780	5.0 P525 V8 CARPATHIAN 110 AUTO 4WD	nan	ESS
CROSSLAND	OPE	01/03/2023	10	6.0	131.0g/km	nan	EDITION	FAUX	CROSSLAND	REU35A02DB	nan	nan	ESS
SERIE 1	BMW	01/05/2020	52948	7.0	135.0g/km	nan	LOUNGE	FAUX	SERIE 1	REU6134C9C	nan	nan	ESS
2008	PEU	01/03/2022	12845	7.0	124.0g/km	5.0	ALLURE	FAUX	2008	REUA241055	1.2 PT 130 S&S ALLURE	nan	ESS

Figure 7 - Extrait d'un fichier brut du partenaire ReezoCar

Dans la figure ci-dessus, nous pouvons constater sur un exemple, et ce malgré le fait qu'il ne présente pas une mauvaise qualité flagrante, que certaines données sont manquantes, ou que d'autres, comme la marque, ne sont pas exploitables directement.

Enfin, la réconciliation par caractéristiques ne répond pas à l'ensemble des besoins opérationnels au sein de CGI Finance. Ainsi, comme vu précédemment dans la présentation des applications, certaines comme Téléfi ou Occazio ont besoin de récupérer les informations provenant du référentiel Argus en n'ayant au départ que la plaque d'immatriculation.

C'est ici qu'intervient la réconciliation par immatriculation.

3.1.3 La réconciliation par immatriculation

La réconciliation par immatriculation est un besoin spécifique de certains clients de CGI Finance (Occazio, Téléfi 3&4) qui, à partir d'une plaque d'immatriculation, doit retrouver les correspondances les plus probables dans le référentiel Argus.

Effectivement, dans le cas d'Occazio, l'application propose à ses utilisateurs de renseigner une plaque d'immatriculation (soit manuellement, soit en la scannant via OCR), et a donc besoin par la suite de retrouver les informations du véhicule dans le référentiel Argus pour pouvoir effectuer ses traitements.

Figure 8 - Exemple de parcours Occazio de la plaque à la finition

Dans la figure ci-contre, un exemple du parcours Occazio avec saisie de plaque d'immatriculation jusqu'à la sélection par l'utilisateur de la finition.

Le besoin de réconciliation est ici d'obtenir une liste de finitions les plus probables, la finition étant la granularité la plus fine à laquelle la réconciliation doit prétendre. Le but étant d'être le plus précis possible afin de n'avoir qu'une finition, ou un minimum de finitions sélectionnées. Afin que l'utilisateur (qui est le grand public dans ce cas) soit guidé au maximum dans le parcours de mise en vente ou d'achat de son véhicule.

Dans le cas de Téléfi 3&4, le besoin est le même, mais avec un public expert (concessionnaire), et dans le but cette fois d'obtenir une simulation de financement au plus proche de la valeur réelle du véhicule, comme nous l'avons défini dans la partie précédente.

Jusqu'à présent, Occazio comme Téléfi 3&4 s'appuyait sur un service externe pour satisfaire ce besoin, que je détaillerai dans la partie sur les solutions préexistantes : le SIV Argus.

Mais pour le moment, maintenant que nous avons compris l'importance de la réconciliation et de ses méthodes, explorons en profondeur le référentiel sur lequel le SI de CGI Finance se base : le référentiel Argus.

3.2 Le référentiel Argus

3.2.1 Présentation générale de l'Argus

Initiée en 1927 par Paul Rousseau, l'Argus s'est imposée comme une entreprise familiale phare du paysage automobile français. Reconnue pour ses cotations de véhicules d'occasion, elle est devenue au fil du temps une référence incontestée pour les professionnels du secteur ainsi que pour les particuliers. Originellement sous forme de journal dédié à la cotation automobile, l'Argus a su évoluer et diversifier ses offres en intégrant des solutions technologiques et des analyses approfondies destinées à l'industrie automobile. [7]

Avec cette solide histoire et cette expertise reconnue, l'Argus ne s'est pas contenté de donner une valeur aux véhicules. Elle a établi un référentiel complet, détaillé et continuellement mis à jour pour répondre aux besoins variés de ses utilisateurs.

3.2.2 Présentation du référentiel Argus

Ce référentiel, au cœur de la stratégie digitale de nombreux acteurs de l'automobile, est la pierre angulaire du SI de CGI Finance. C'est une source d'information riche, qui permet de définir rigoureusement un véhicule : sa génération, sa motorisation, sa finition commerciale, son prix, ses caractéristiques techniques, ses équipements de série et l'intégralité des options, couleurs et selleries disponibles. [8]

Dans le cas de CGI Finance, ce référentiel se présente sous la forme d'une base de données répartie en plusieurs tables. Nous allons à présent nous pencher sur sa structure.

3.2.3 Structure du référentiel Argus chez CGI Finance

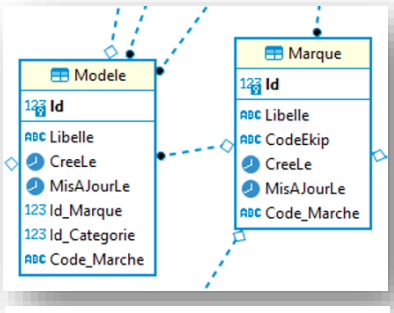


Figure 9 - Aperçu du diagramme du référentiel Argus

Voir Annexe 3.2.3

CGI Finance s'appuie sur les données du référentiel Argus, importées via des fichiers CSV. Une fois importées, ces données sont intégrées dans une base de données PostgreSQL, puis enrichies des spécificités propres à CGI Finance. Pour des raisons de pertinence, nous ne détaillerons pas ces spécificités ici, mais il est à noter qu'elles concernent principalement les tables préfixées par « Ekip », ainsi que les champs portant ce mot en suffixe (pour plus de détails, se référer à l'Annexe 3.2.3 qui présente le diagramme complet de la base).

La structure de cette base est certes complexe, mais elle reflète la richesse et la diversité des données du référentiel Argus. Bien que nous n'ayons pas la possibilité de modifier cette structure, il est essentiel de la comprendre pour optimiser son utilisation.

L'élément central de cette structure est la table « Version ». Non seulement elle offre la granularité la plus fine, mais elle présente aussi des subtilités. Un exemple notable est la présence de doublons de finitions. Ce que nous allons creuser davantage dans la section suivante.

Précision importante : CGI Finance ne finançant que des véhicules avec au maximum 10 ans d'ancienneté, la base disponible a pour finitions les plus anciennes celles dont les dates de commercialisation sont aux alentours de l'année 2010.

3.2.4 Subtilités importantes de la table « Version »

Comme mentionné plus haut, le but ultime de la réconciliation est de trouver une finition particulière (ou la liste des finitions les plus probables) suivant les caractéristiques de véhicules en entrée (ou des caractéristiques récupérées par la plaque d'immatriculation).

Le référentiel Argus servant de point de comparaison, une étude approfondie de celui-ci était donc indispensable, et il s'est avéré qu'il y avait des subtilités particulières à prendre en compte, et ce avant même le début de la conception de l'application.

3.2.4.1 Petite clarification lexicale

Dans le référentiel, ce que l'on appelle couramment "finition" est représenté par le champ « idexterne » de la table « Version ». Pour des raisons de simplicité, et étant donné que ce terme est fréquemment utilisé par les acteurs de CGI Finance pour désigner une finition, nous le nommerons « Id Argus » dans la suite de ce document. De plus, il est important de souligner que le terme « finition » sera utilisé plus tard pour décrire un concept légèrement différent dans le contexte technique.

Cette clarification terminologique est cruciale dès maintenant en raison des divergences et incompréhensions rencontrées au cours de cette mission. En effet, le même terme n'a pas toujours la même signification pour les équipes métier et techniques. Par exemple, alors que les équipes métier utilisent le terme "finition" (ou même "version") pour désigner un véhicule spécifique, l'équipe technique préfère le terme "Id Argus" pour éliminer toute ambigüité.

Nous approfondirons cette terminologie dans la section consacrée à la conception.

Cependant, une clarification à ce stade était nécessaire, car nous sommes en pleine transition des notions métier vers les notions techniques dans l'état de l'art.

3.2.4.2 Les « doublons » d'id argus dans le référentiel

Une fois ces clarifications terminologiques faites, nous pouvons nous attaquer à la plus importante spécificité de la table « Version » du référentiel.

En effet, suite à des études préliminaires, il s'est avéré qu'il existait plusieurs lignes pour la plupart des id Argus dans la table « Version », sachant que trouver un ou plusieurs id Argus comme résultat de la réconciliation est le but, il fallait comprendre pourquoi il existait de tels doublons dans la base de référence.

nb_id_argus	nb_lignes_total	moyennes_occurences_id_argus	nb_minimal_id_argus	nb_maximal_id_argus	ecart_type_occurences_id_argus
124 799	318 320	2,5506614636	1	16	1,2255377116

Figure 10 - Analyse quantitative de la répartition des id argus dans la table Version

Voir Annexe 3.2.4.2

L'analyse quantitative de la table « Version » révèle certains aspects intéressants de la répartition des id Argus. Chaque id Argus apparaît en moyenne 2,55 fois. Cette valeur centrale est complétée par une dispersion, représentée par un écart-type de 1,22. Cela signifie que, bien que la tendance centrale soit de 2,55 occurrences, une grande partie des id Argus varie dans un intervalle allant de 1,33 à 3,77 occurrences.

L'existence d'une valeur minimale de 1 indique qu'il y a des id Argus qui sont uniques dans la table. Par contre, la présence d'une valeur maximale de 16 suggère qu'il existe des id Argus qui sont bien plus fréquents que la moyenne, se distinguant nettement de la tendance générale.

Ces statistiques offrent une vue d'ensemble sur la distribution (voir figure 12 ci-dessous) des id Argus, montrant à la fois la tendance générale et les variations notables, néanmoins il faut se pencher sur la raison de l'existence de tels doublons, et voir si, selon les besoins, cela pourrait poser problème par la suite.

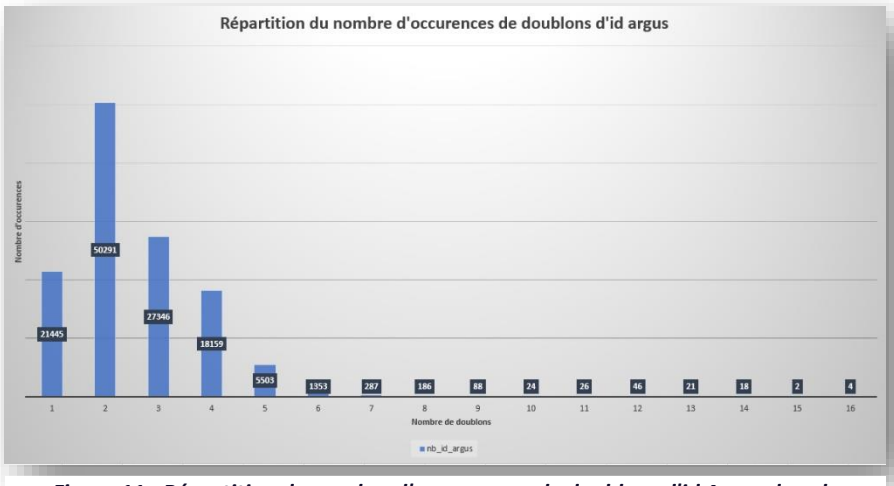
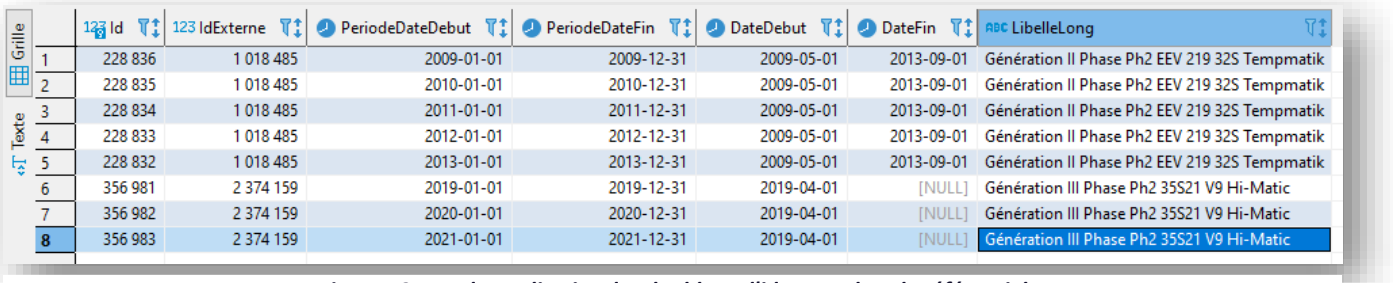


Figure 11 - Répartition du nombre d'occurrences de doublons d'id Argus dans le référentiel

3.2.4.3 La raison des « doublons » d'id Argus

En effectuant une analyse plus approfondie au niveau de la table « Version », nous pouvons clarifier la structure de cette table et la raison de ces doublons.



	Id	IdExterne	PeriodeDateDebut	PeriodeDateFin	DateDebut	DateFin	LibelleLong
1	228 836	1 018 485	2009-01-01	2009-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
2	228 835	1 018 485	2010-01-01	2010-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
3	228 834	1 018 485	2011-01-01	2011-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
4	228 833	1 018 485	2012-01-01	2012-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
5	228 832	1 018 485	2013-01-01	2013-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
6	356 981	2 374 159	2019-01-01	2019-12-31	2019-04-01	[NULL]	Génération III Phase Ph2 35S21 V9 Hi-Matic
7	356 982	2 374 159	2020-01-01	2020-12-31	2019-04-01	[NULL]	Génération III Phase Ph2 35S21 V9 Hi-Matic
8	356 983	2 374 159	2021-01-01	2021-12-31	2019-04-01	[NULL]	Génération III Phase Ph2 35S21 V9 Hi-Matic

Figure 12 - Etude qualitative des doublons d'id Argus dans le référentiel

Voir Annexe 3.2.4.3

En analysant l'annexe 3.2.4.3, nous avons deux exemples de véhicules avec plusieurs entrées identiques d'id Argus.

D'après cet exemple, nous pouvons en déduire :

- Que pour garantir l'unicité de chaque enregistrement dans la table il y a un champ clé primaire nommé « Id ».
- Qu'en analysant le champ « LibelleLong », que les doublons correspondent bel et bien à une seule et même finition dont les lignes ont été dupliquées.
- Qu'elles ont toutes la même valeur (pour un même id Argus bien évidemment) de « DateDebut » et « DateFin ».
- Que les seules valeurs qui diffèrent sont « PeriodeDateDebut » et « PeriodeDateFin ».
- Que les valeurs les plus anciennes de « PeriodeDateDebut » correspond à « DateDebut » et que logiquement les valeurs les plus récentes de « PeriodeDateFin » correspondent à « DateFin », à l'exception faite quand « DateFin » est nulle.
- Que les valeurs de « PeriodeDateDebut » et « PeriodeDateFin » sur une même ligne pour un même id Argus représentent un intervalle d'un an.

Ainsi, d'après ces observations, et ce parce que nous n'avions pas le « mode d'emploi » du référentiel, nous avons pu déduire plusieurs points importants de son fonctionnement qui ont été confirmés plus tard par des informations métier:

- Que l'unicité dans le référentiel n'est pas déterminée par l'id Argus, ce qui démontre un niveau de granularité encore plus fin.
- Que les champs « DateDebut » et « DateFin » correspondent respectivement aux dates de début et de fin de commercialisation de la finition ayant cet id Argus.

- Que les champs « PeriodeDateDebut » et « PeriodeDateFin » correspondent à des subdivisions annuelles entre la date de début et de fin de commercialisation de la finition en question. D'autant plus que certaines informations changent suivant l'année de commercialisation, mais nous reviendrons sur ce point dans les prochaines sections sur la qualité des données du référentiel.

Ainsi, sur la base de ces analyses, la table « Version » est plus complexe que ce que l'on aurait pu penser de prime abord. En effet en elle résident deux granularités différentes que sont :

- Une granularité au niveau « id Argus » qui est celle qui nous intéresse pour aboutir au résultat demandé par le métier.
- Une granularité encore plus fine, au niveau des années de commercialisation d'un unique id Argus, qui peut poser problème de par la fluctuation de certaines informations d'une année sur l'autre. Variations ne faisant pas forcément partie de la table « Version », et qui feront l'objet des prochaines sections.

3.2.5 Fluctuation du prix catalogue au fil des années

Le processus de réconciliation visant à déterminer un id Argus spécifique pour chaque finition de véhicule implique de comprendre l'évolution du prix catalogue (PrixTTC) au fil du temps. Nous avons observé que le prix catalogue peut varier significativement pour une même finition au cours des années.

	Id integer	IdExterne numeric	PeriodeDateDebut date	PeriodeDateFin date	DateDebut date	DateFin date	ConsommationEmissionCo2 numeric	LibelleLong character varying (200)	PrixTTC numeric
1	176446	1084508	2014-01-01	2014-12-31	2014-04-24	2016-07-28	174	Génération I Phase Ph2 3.0 V6 TFSI 272ch S line quattro S tronic 7 Euro6	58710.00
2	176445	1084508	2015-01-01	2015-12-31	2014-04-24	2016-07-28	179	Génération I Phase Ph2 3.0 V6 TFSI 272ch S line quattro S tronic 7 Euro6	59280.00
3	176444	1084508	2016-01-01	2016-12-31	2014-04-24	2016-07-28	179	Génération I Phase Ph2 3.0 V6 TFSI 272ch S line quattro S tronic 7 Euro6	59280.00
4	50529	1084517	2014-01-01	2014-12-31	2014-04-24	2016-07-28	174	Génération I Phase Ph2 3.0 V6 TFSI 272ch Avus quattro S tronic 7 Euro6	63360.00
5	50528	1084517	2015-01-01	2015-12-31	2014-04-24	2016-07-28	179	Génération I Phase Ph2 3.0 V6 TFSI 272ch Avus quattro S tronic 7 Euro6	63930.00
6	50527	1084517	2016-01-01	2016-12-31	2014-04-24	2016-07-28	179	Génération I Phase Ph2 3.0 V6 TFSI 272ch Avus quattro S tronic 7 Euro6	63930.00
7	3849	1153405	2014-01-01	2014-12-31	2014-03-13	2018-02-13	114	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendline	20700.00
8	3850	1153405	2015-01-01	2015-12-31	2014-03-13	2018-02-13	114	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendline	20990.00
9	3851	1153405	2016-01-01	2016-12-31	2014-03-13	2018-02-13	116	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendline	21520.00
10	3852	1153405	2017-01-01	2017-12-31	2014-03-13	2018-02-13	116	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendline	21520.00
11	3853	1153405	2018-01-01	2018-12-31	2014-03-13	2018-02-13	116	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendline	21520.00
12	93511	1176775	2014-01-01	2014-12-31	2014-06-02	2016-06-01	117	Génération II Phase Ph1 BlueHDi 150ch Intensive S&S EAT6	33950.00
13	93512	1176775	2015-01-01	2015-12-31	2014-06-02	2016-06-01	112	Génération II Phase Ph1 BlueHDi 150ch Intensive S&S EAT6	33850.00
14	93513	1176775	2016-01-01	2016-12-31	2014-06-02	2016-06-01	112	Génération II Phase Ph1 BlueHDi 150ch Intensive S&S EAT6	33850.00
15	69875	1176777	2014-01-01	2014-12-31	2014-06-02	2016-06-01	117	Génération II Phase Ph1 BlueHDi 150ch Business + S&S EAT6	34200.00
16	69876	1176777	2015-01-01	2015-12-31	2014-06-02	2016-06-01	112	Génération II Phase Ph1 BlueHDi 150ch Business + S&S EAT6	34400.00
17	69877	1176777	2016-01-01	2016-12-31	2014-06-02	2016-06-01	112	Génération II Phase Ph1 BlueHDi 150ch Business + S&S EAT6	34400.00

Figure 13 - Fluctuation du prix catalogue en fonction des années

Ces variations reflètent les ajustements effectués par les constructeurs automobiles, qui modifient régulièrement les prix neufs de leurs modèles. Ces modifications tarifaires peuvent se traduire par une augmentation, une diminution ou une stabilité des prix au cours de la période de commercialisation d'une finition donnée, et sont liées à des facteurs souvent externes comme l'inflation, l'augmentation des matières premières etc…

Cette dynamique tarifaire est un élément qui semble crucial à prendre en compte dans notre démarche de réconciliation, afin d'assurer la précision et la pertinence des prix de référence utilisés pour les analyses et décisions financières.

3.2.6 Fluctuation de l'émission CO2 au fil des années

De manière similaire à la fluctuation des prix catalogue, nous observons également des variations annuelles dans d'autres données, notamment les émissions de CO2. Cette variabilité se manifeste de deux façons distinctes : absence de données et fluctuation des valeurs (voir figure 15).

	Id [PK] integer	IdExterne numeric	PeriodeDateDebut date	PeriodeDateFin date	DateDebut date	DateFin date	ConsommationEmissionCo2 numeric	LibelleLong character varying (200)
1	141501	1088702	2011-01-01	2011-12-31	2011-01-01	2014-03-17	[null]	Génération II Phase Ph2 Euro5 30 L2H2 HDi 110 Club
2	141500	1088702	2012-01-01	2012-12-31	2011-01-01	2014-03-17	[null]	Génération II Phase Ph2 Euro5 30 L2H2 HDi 110 Club
3	141499	1088702	2013-01-01	2013-12-31	2011-01-01	2014-03-17	203	Génération II Phase Ph2 Euro5 30 L2H2 HDi 110 Club
4	141498	1088702	2014-01-01	2014-12-31	2011-01-01	2014-03-17	203	Génération II Phase Ph2 Euro5 30 L2H2 HDi 110 Club
5	3849	1153405	2014-01-01	2014-12-31	2014-03-13	2018-02-13	114	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendli...
6	3850	1153405	2015-01-01	2015-12-31	2014-03-13	2018-02-13	114	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendli...
7	3851	1153405	2016-01-01	2016-12-31	2014-03-13	2018-02-13	116	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendli...
8	3852	1153405	2017-01-01	2017-12-31	2014-03-13	2018-02-13	116	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendli...
9	3853	1153405	2018-01-01	2018-12-31	2014-03-13	2018-02-13	116	Génération I Phase Ph1 1.2 TSI 85ch BlueMotion Technology Trendli...

Figure 14 - Fluctuation des émission CO2 en fonction des années

Pour les cas où les données sur les émissions de CO2 sont manquantes, il ne semble pas exister de justification métier pour cette absence. Étant donné qu'un véhicule (à l'exception des véhicules électriques) émet toujours du CO2, cette absence indique un problème de qualité des données dans le référentiel.

Quant aux fluctuations observées dans les émissions de CO2, elles sont souvent en hausse. Un exemple significatif concerne un modèle Volkswagen dont la période de commercialisation coïncide avec le scandale du DieselGate en 2015. Ce scandale, bien que principalement axé sur la minimisation des émissions de Nox, a également mis en lumière des problèmes relatifs aux émissions de CO2 [9]. Ces variations, même si elles sont minimes, sont importantes à considérer pour la conception d'un système de réconciliation fiable et précis. Elles reflètent non seulement les ajustements techniques apportés aux véhicules mais aussi les impacts des événements extérieurs sur les données du référentiel.

En résumé, la compréhension et la prise en compte de ces fluctuations sont essentielles pour garantir l'intégrité et la pertinence des données utilisées dans nos analyses et décisions.

3.2.7 Conclusion sur le référentiel Argus

En conclusion, l'analyse du référentiel Argus révèle son importance capitale comme base de comparaison pour la réconciliation des données chez CGI Finance. Ce référentiel, riche et complexe, est essentiel pour une réconciliation précise, comme le montre la diversité des données dans la table Version et les nuances liées aux doublons d'Id Argus. Ces éléments accentuent l'importance d'une compréhension détaillée du référentiel pour un processus de réconciliation efficace.

Il est cependant crucial de reconnaître que le référentiel Argus, tout en étant une source de données exhaustive, présente certains défis, notamment en matière de qualité des données, comme l'illustrent des cas de données manquantes. Par contre, la variabilité observée dans certaines données, telles que les émissions de CO2 et les prix catalogue, loin d'être un défaut, témoigne de la précision et de l'adaptabilité du référentiel aux évolutions du marché. Cette variabilité reflète les dynamiques du marché automobile et les ajustements des constructeurs, fournissant ainsi une image réaliste et à jour du secteur.

Bien que le prix et les émissions de CO2 soient des exemples spécifiques et non exhaustifs, ils illustrent la nécessité d'une vigilance et d'une adaptation continues face à la complexité et à l'évolution constante des données du marché. Cette analyse approfondie du référentiel Argus confirme sa valeur inestimable en tant qu'outil de comparaison pour CGI Finance. Elle souligne également la nécessité d'une gestion minutieuse et d'une analyse rigoureuse des données pour assurer l'intégrité et la pertinence des informations utilisées, ouvrant la voie à des améliorations continues des processus de réconciliation pour s'aligner sur les besoins en évolution de l'entreprise dans un secteur en constante mutation.

3.3 Le Service d'Immatriculation des Véhicules

3.3.1 Présentation générale du S.I.V.

Le Service d'Immatriculation des Véhicules (S.I.V.) en France est un système centralisé pour la gestion des immatriculations de véhicules. Introduit en 2009, il a remplacé le précédent système d'immatriculation datant de 1950. Le S.I.V. attribue un numéro d'immatriculation unique à vie pour chaque véhicule, le conservant jusqu'à sa destruction ou son exportation. Ce numéro est composé de sept caractères alphanumériques, attribués chronologiquement dans une série nationale unique, et est impossible à choisir. Le nouveau système a été mis en place pour remplacer l'ancien système vieillissant, offrant une gestion plus efficace et plus sûre des immatriculations, et permettant l'adaptation à l'administration électronique.

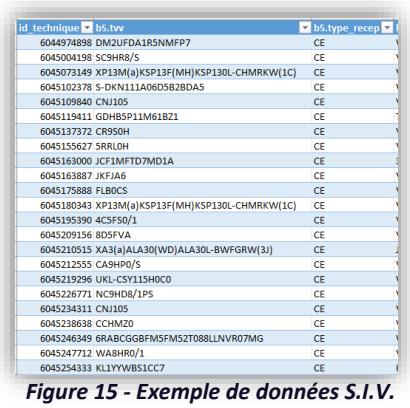
Le S.I.V. a été introduit suite à la constatation du vieillissement du système de gestion des cartes grises et des serveurs informatiques. La nécessité d'anticiper l'expiration du dispositif de numérotation datant de 1950, qui allait rapidement atteindre ses limites, en particulier à Paris, a été un facteur clé dans son développement. La réflexion sur le remplacement du système d'immatriculation a commencé dès 2001, impliquant le ministère chargé des transports, la profession automobile, et d'autres ministères concernés. [10]

3.3.2 Les données du S.I.V.

Dans le cadre du projet, CGI Finance a acquis les données du S.I.V. depuis 2008, dans le but de pouvoir effectuer des réconciliation par immatriculation.

Ces données, en l'état actuel, sont sous la forme de fichiers CSV à récupérer sur un serveur sécurisé du ministère. Elles comprennent :

- Un fichier de base
- Un fichier de mise à jour disponible tous les vendredis



id_technique	b5.tvv	b5.type_recep
6044974898	DM2UFDA1R5NMFP7	CE
6045004198	SC9HR8/S	CE
6045073149	XP13M(a)KSP13F(MH)KSP130L-CHMRKW(1C)	CE
6045102378	S-DKN111A06D5B2BDA5	CE
6045109840	CNJ105	CE
6045119411	GDHB5P11M61BZ1	CE
6045137372	CR9S0H	CE
6045155627	5RRL0H	CE
6045163000	JCF1MFTD7MD1A	CE
6045163887	JKFJA6	CE
6045175888	FLB0CS	CE
6045180343	XP13M(a)KSP13F(MH)KSP130L-CHMRKW(1C)	CE
6045195390	4C5FS0/1	CE
6045209156	8D5FVA	CE
6045210515	XA3(a)ALA30(WD)ALA30L-BWFGRW(3J)	CE
6045212555	CA9HP0/S	CE
6045219296	UKL-CSY115H0C0	CE
6045226771	NC9HD8/1PS	CE
6045234311	CNJ105	CE
6045238638	CCHMZ0	CE
6045246349	6RABCGGBFM5FM52T088LLNVR07MG	CE
6045247712	WA8HR0/1	CE
6045254333	KL1YYWBS1CC7	CE

Figure 15 - Exemple de données S.I.V.

Voir Annexe 3.3.2

Ces données comprennent toutes les données carte grise de chaque Véhicule Particulier (VP) immatriculé en France, hormis les données personnelles des propriétaires dont nous n'avons pas l'utilité dans le cadre de notre projet.

Ces données brutes seront par ailleurs difficilement utilisables telles quelles dans un processus de réconciliation, il est donc à ce stade envisagé de les importer de manière automatique dans une base de données.

Il est important à savoir qu'il existe trois types d'immatriculations :

- Les immatriculation définitives récentes (format XX-000-XX)
- Les immatriculations définitives anciennes (format 00(0)-XX(X)-00)
- Les immatriculations provisoires (format WW-000-XX)

Les immatriculations provisoires sont destinées aux véhicules nouvellement immatriculés, jusqu'à l'attribution des immatriculations définitives. Celles-ci sont attribuées de façon cyclique, et donc peuvent être attribuées à plusieurs véhicules dans ces données, contrairement aux immatriculations définitives qui, elles, correspondent toujours à un seul et même véhicule.

3.4 Les solutions existantes

3.4.1 La réconciliation dans Téléfi

La solution Téléfi intègre un algorithme de réconciliation d'Id Argus relativement basique. Bien que le code spécifique de cet algorithme ne nous ait pas été accessible pour une analyse détaillée, nous comprenons qu'il s'appuie sur des correspondances précises trouvées dans le référentiel Argus pour chaque type de données. Cette méthode s'avère efficace lorsque les données traitées sont complètes et formatées correctement. Cependant, sa performance diminue notablement en présence de données imparfaites ou mal formatées.

Un autre inconvénient notable de cette solution est son intégration profonde dans l'architecture de Téléfi, ce qui limite sa capacité à être employée pour d'autres applications ou clients au sein de CGI Finance. Cette limitation a mis en évidence le besoin de développer une nouvelle approche, à la fois indépendante et améliorée, pour répondre aux exigences de précision et de flexibilité dans le processus de réconciliation des données.

Néanmoins, les résultats de cet algorithme pourra nous servir de base de comparaison avec la solution que nous allons développer, fournissant ainsi un cadre d'évaluation.

3.4.2 La réconciliation dans Occazio

Le service de réconciliation en Python au sein de l'application Occazio utilise une approche basique mais essentielle pour rapprocher les données des véhicules avec les ID Argus. Bien que constituant une partie de l'ensemble de l'application principalement développée en C#, ce service se distingue par son approche méthodique.

Un aspect crucial de ce service est le nettoyage des données, une étape préparatoire nécessaire pour assurer la clarté et la cohérence des informations, comme ici avec le nettoyage de la puissance DIN :

```python
def nettoyage_puissance_din(self, dataframe, puiss_din):
    """ Nettoie le libellé de la puissande DIN """
    if puiss_din is not None:

        rng = range(int(puiss_din*(1 + self.puissance_din_window/100)),
                    int(puiss_din*(1 - self.puissance_din_window/100 + 1)))

        for p in rng:

            dataframe.Field_compared_LibTech = dataframe.Field_compared_LibTech.str.replace(str(p), "")
            dataframe.Field_compared_LibTech = dataframe.Field_compared_LibTech.str.replace("ch ", "")

            dataframe.Field_compared_LibLong = dataframe.Field_compared_LibLong.str.replace(str(p), "")
            dataframe.Field_compared_LibLong = dataframe.Field_compared_LibLong.str.replace("ch ", "")

            self.dico["Version_modifie"] = self.dico["Version_modifie"].replace(str(p), "").replace("ch ", "")
            self.dico["Titre_annonce_modifie"] = self.dico["Titre_annonce_modifie"].replace(str(p), "").replace(
                "ch ", "")

            if self.dico["Description_annonce_modifie"] is not None:
                self.dico["Description_annonce_modifie"] = self.dico["Description_annonce_modifie"] \
                    .replace(str(p), "") \
                    .replace("ch ", "")
```

Figure 16 – Exemple de fonction de nettoyage de la réconciliation Occazio

```python
def execute(self) -> RapprochementIdArgusOut:
    """ Effectue le rapprochement entre le dictionnaire et une liste d'id argus.
    Gère les différents cas d'erreurs """
    try:
        ids_argus_rapproches = self.look_for_most_scoring_ids()
    except RapprochementException as e:
        rapprochement = RapprochementIdArgusOut(
            succes=False,
            ids_argus=[]
        )

        if isinstance(e, UnknownEnergyException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.energie_inconnue
        elif isinstance(e, UnknownBrandException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.marque_inconnue
        elif isinstance(e, UnknownModelException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.modele_inconnu
        elif isinstance(e, UnknownYearException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.annee_inconnue
        elif isinstance(e, VehicleTooOldException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.vehicule_trop_vieux
        elif isinstance(e, NoResultWithFirstQueryException) \
                or isinstance(e, NoResultWhenFilteringByPositiveProbability) \
                or isinstance(e, NoResultWhenFilteringByPositiveProbability):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.aucun_vehicule_matche

        return rapprochement
```

Figure 17 – Extrait de la fonction de classement

Voir Annexe 3.4.2

Après le nettoyage, le service procède au rapprochement proprement dit, en employant un modèle probabiliste pour évaluer la pertinence des correspondances entre les données du véhicule et les ID Argus. Cela implique le calcul de scores ou de probabilités pour classer les correspondances selon leur degré de pertinence, comme le montre cet autre extrait ci-contre et dans l'annexe 3.3.2.

Il est important de noter que, malgré son rôle dans le rapprochement des données, le service de réconciliation Python d'Occazio présentait certaines limitations dans son algorithme, le rendant moins fiable pour des cas plus complexes.

En conséquence, et afin de répondre aux besoins des autres clients de CGI Finance ainsi qu'à une stratégie d'harmonisation des technologies Back-End essentiellement en .NET, le développement d'un nouveau service de réconciliation indépendant et amélioré, utilisable non seulement par Occazio mais également par d'autres applications, s'est avéré nécessaire.

3.4.3 La réconciliation par immatriculation proposée par l'Argus

Dans le contexte actuel, la réconciliation de véhicules avec le référentiel Argus a souvent impliqué l'usage de l'API de réconciliation par immatriculation offerte par l'Argus [11]. Cette solution, reconnue pour sa précision et sa fiabilité, s'est avérée être un outil indispensable dans le domaine de l'automobile, facilitant l'accès rapide à des données détaillées de véhicules à partir de leur numéro d'immatriculation.

Cependant, malgré son efficacité, le coût de cette solution externe s'est révélé être une considération importante. Avec un prix unitaire par requête estimé à environ 0,20 euros, les coûts mensuels pour une utilisation intensive s'élèvent rapidement, atteignant approximativement 60 000 euros par mois pour CGI Finance.

Face à cet investissement financier significatif, il a été décidé de développer en interne une solution alternative de réconciliation par immatriculation. Cette décision s'inscrit dans une démarche d'optimisation des coûts et d'adaptation aux besoins spécifiques de nos clients. En développant notre propre solution, nous visons à réduire les dépenses tout en maintenant, voire en améliorant, la qualité et l'efficacité du service de réconciliation.

3.5 Conclusion de l'état de l'art

L'analyse approfondie des besoins métiers et des solutions existantes dans le domaine de la réconciliation de données de véhicules a mis en lumière l'importance critique d'un processus précis et fiable. Le référentiel Argus, avec ses subtilités et sa complexité, se révèle être une pierre angulaire pour une réconciliation efficace, mais aussi un défi en termes de gestion et d'interprétation des données.

Les solutions existantes, bien qu'efficaces dans certains contextes, ont révélé des limitations, notamment en termes de coût et de flexibilité. L'utilisation de l'API de réconciliation par immatriculation de l'Argus, par exemple, bien que précise, s'est avérée coûteuse, entraînant la nécessité d'une alternative interne.

Cette analyse nous a permis de comprendre en profondeur le contexte dans lequel s'inscrit notre projet et de définir les axes d'amélioration pour notre solution de réconciliation. En tenant compte des défis identifiés, nous visons à développer une solution qui non seulement répond aux exigences de précision et d'efficience, mais qui est également économiquement viable et bien intégrée dans l'écosystème de CGI Finance.

Ce travail préparatoire constitue donc un fondement solide pour la conception et le développement d'une solution de réconciliation robuste et sur mesure, qui répondra aux besoins spécifiques de nos clients et s'alignera sur les objectifs stratégiques de CGI Finance.

4 CONCEPTION

4.1 Terminologie technique

Alors que nous nous apprêtons à aborder la phase de conception, il est essentiel de s'assurer d'une compréhension commune des termes techniques qui seront employés. Comme nous l'avons constaté dans les sections précédentes, certains termes peuvent prêter à confusion, notamment lorsqu'ils sont interprétés différemment selon qu'ils sont utilisés dans un contexte métier ou technique. Afin de prévenir toute ambiguïté et de faciliter la compréhension, nous allons établir ici une corrélation entre les diverses terminologies employées, en fonction de leur contexte d'utilisation.

Il est important de noter que, dans la suite de ce document, nous adopterons une perspective principalement technique pour ces termes.

Vous pourrez retrouver le tableau associé en Annexe 4.1.

4.2 Concepts théoriques

4.2.1 La notion de dimension

Pour entamer efficacement notre phase de conception, nous débutons par une étape cruciale : la segmentation des caractéristiques des véhicules en « dimensions ». Chaque dimension correspond à une caractéristique spécifique d'un véhicule, définie par son type et sa valeur. Cette approche méthodique permet une comparaison structurée et individualisée de chaque caractéristique d'un véhicule avec les données du référentiel Argus.

Cette segmentation en dimensions est essentielle pour plusieurs raisons. Premièrement, elle élimine les risques de confusion ou d'erreur en comparant des caractéristiques homogènes et directement pertinentes. Deuxièmement, et c'est un point fondamental, elle établit la base nécessaire pour calculer des distances entre les caractéristiques des données des DMS et celles du référentiel Argus. En évaluant ces distances, nous pouvons quantifier la similitude ou la divergence entre un véhicule donné et les modèles référencés dans l'Argus, ce qui est crucial pour le processus de réconciliation.

Pour identifier les dimensions pertinentes, une analyse approfondie des données du référentiel Argus et des DMS a été menée, révélant des corrélations essentielles pour notre démarche. Des exemples tirés de DMS, notamment de ReezoCar et Planète VO, sont présentés dans les annexes 4.2.1.a et 4.2.1.b.

La liste des dimensions, résultant de cette étude minutieuse, forme le socle de notre processus de réconciliation :

- Marque
- Modèle
- Sous-Modèle
- Génération
- Finition
- Phase
- Version
- Energie
- Boite de Vitesse
- Type Boite de Vitesse
- Carrosserie
- Nombre de Places
- Nombre de Portes
- Puissance Fiscale
- Puissance DIN
- Cylindrée
- Consommation
- Poids
- Prix Catalogue
- Date Première MEC
- Emission CO2

La définition précise de ces dimensions est le fondement sur lequel repose le calcul des distances entre les caractéristiques des véhicules et celles du référentiel Argus, une étape clé qui sera approfondie dans les sections suivantes.

4.2.2 Les distances : principe général

Suite à l'établissement des dimensions des véhicules, tant pour ceux à analyser que pour les candidats issus du référentiel Argus, notre prochaine étape consiste à comparer ces dimensions. Pour cela, nous avons adopté une approche basée sur le calcul de distances. Cette méthode évalue la similitude entre chaque dimension d'un véhicule et celles des candidats potentiels dans le référentiel. Dans ce système, la « distance » est une valeur numérique allant de 0 à 1 attribuée à chaque dimension du véhicule candidat. Un score de 0 indique une correspondance parfaite entre les dimensions comparées, tandis qu'une valeur s'approchant de 1 signifie une divergence croissante entre les caractéristiques.

De manière significative, chaque type de dimension se voit attribuer une pondération spécifique. Cette pondération joue un rôle crucial dans le calcul de la distance générale entre les véhicules, permettant de prioriser certaines caractéristiques sur d'autres. En effet, certaines dimensions peuvent être jugées plus critiques que d'autres dans l'évaluation de la correspondance d'un véhicule. Cette flexibilité dans la pondération offre également la possibilité d'ajuster les résultats en fonction des exigences spécifiques de différents partenaires.

Ainsi, l'utilisation des distances et des pondérations s'avère être un outil puissant pour quantifier la similitude entre les véhicules, formant une base solide pour le classement et la sélection des correspondances les plus pertinentes dans notre processus de réconciliation.

4.2.3 Distance discrète : Evaluation de l'égalité parfaite

Dans notre processus de réconciliation, la 'distance discrète' est la première méthode utilisée pour évaluer la similitude entre les dimensions. Elle repose sur la vérification stricte de l'égalité entre la dimension à analyser d'un véhicule et celle correspondante du candidat dans le référentiel Argus.

La formule mathématique de la distance discrète est la suivante :

$$d(v_{test}, v_{ref}) = \begin{cases} 0, si\ v_{ref} = v_{test} \\ 1, \qquad sinon \end{cases}$$

$v_{test} = valeur\ de\ la\ dimension$ à $tester$

$v_{ref} = valeur\ de\ la\ dimension\ venant\ du\ référentiel$

Si les valeurs de la dimension comparée sont identiques (par exemple, même marque ou même modèle), la distance discrète est de 0, indiquant une correspondance parfaite. En revanche, si les valeurs ne correspondent pas, la distance est fixée à 1, reflétant une absence de similitude totale pour cette dimension spécifique. Il est de plus à noter que ce calcul de distance fonctionne aussi bien sur les valeurs numériques que textuelles.

Cette méthode de calcul de la distance est particulièrement utile pour les caractéristiques qui requièrent une correspondance exacte, offrant ainsi un moyen rapide et clair de déterminer si certaines dimensions clés sont identiques entre le véhicule analysé et les candidats potentiels.

4.2.4 Distance Euclidienne normalisée

Pour les dimensions numériques, telles que le nombre de portes, le nombre de places, les puissances fiscales et DIN, nous appliquons une version normalisée de la distance euclidienne. Cette approche est spécialement conçue pour comparer des valeurs numériques au sein de dimensions homogènes.

La formule de calcul est la suivante :

$$d(v_{test}, v_{ref}) = \frac{|v_{test} - v_{ref}|}{max(v_{test}, v_{ref})}$$

$v_{test} = valeur\ de\ la\ dimension$ à $tester$
$v_{ref} = valeur\ de\ la\ dimension\ venant\ du\ référentiel$

Cette normalisation garantit que la distance calculée est relative à la plus grande des deux valeurs, permettant ainsi une comparaison cohérente et proportionnelle des caractéristiques.

Pour illustrer, considérons le calcul de la distance entre 2 et 4 pour une dimension donnée, par exemple, la puissance fiscale :

$$d(2,4) = \frac{2}{4} = 0.5$$

Cette distance de 0.5 indique une correspondance à mi-chemin entre une adéquation parfaite et une différence maximale, selon les valeurs comparées. Cette méthode est particulièrement utile pour évaluer la similitude entre les données, offrant un moyen simple et efficace de mesurer les écarts entre les caractéristiques des véhicules dans notre processus de réconciliation.

4.2.5 Distance de Levenshtein normalisée

Lors de l'analyse des dimensions textuelles telles que la finition ou la carrosserie des véhicules, nous utilisons la distance de Levenshtein pour mesurer la similitude entre les chaînes de caractères. Cette méthode calcule le nombre minimal de modifications (insertions, suppressions, substitutions) nécessaires pour transformer une chaîne de caractères en une autre.

La formule de la distance de Levenshtein entre deux chaînes est définie comme suit :

$$d(v_{test}, v_{ref}) = \begin{cases} \|v_{test}\| & si\ \|v_{ref}\| = 0 \\ \|v_{ref}\| & si\ \|v_{test}\| = 0 \\ d(v'_{test}, v'_{ref}) & si\ v_{ref}[0] = v_{test}[0] \\ 1 + min \begin{cases} d(v'_{test}, v_{ref}) \\ d(v_{test}, v'_{ref}) \\ d(v'_{test}, v'_{ref}) \end{cases} & sinon \end{cases}$$

$Avec$:

$\|v_{test}\|$ $la\ taille\ de\ la\ chaine$

v'_{test} $la\ chaine\ v_{test}\ privée\ de\ son\ premier\ caractère$

Pour rendre cette mesure plus adaptée à notre contexte et permettre une comparaison équitable entre chaînes de longueurs différentes, nous normalisons cette distance. La distance normalisée est calculée en divisant le nombre de modifications par le nombre maximum de caractères entre les deux chaînes comparées, ce qui de plus donnera une valeur entre 0 et 1 :

$$Distance\ normalisée = \frac{d(v_{test}, v_{ref})}{\max(\|v_{test}\|, \|v_{ref}\|)}$$

Cette méthode normalisée est essentielle dans notre processus de réconciliation pour comparer les données textuelles, car elle fournit une évaluation précise et adaptée de la similitude, tenant compte des petites variations ou des erreurs dans les chaines de caractères.

4.2.6 Distance de Jaccard normalisée

Pour les dimensions textuelles complexes, comme les libellés qui contiennent plusieurs mots, nous avons recours à une version adaptée de la distance de Jaccard. Cette mesure est particulièrement utile pour les chaînes de mots longues et complexes. La distance de Jaccard traditionnelle calcule la dissimilarité entre deux ensembles, et se définit comme le rapport entre la taille de leur intersection et la taille de leur union :

$$d_{jaccard}(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

$$o\text{ù } A \text{ et } B \text{ sont des ensembles de mots}$$

Dans notre adaptation de cette formule pour les besoins spécifiques du projet, nous calculons la distance en se concentrant sur le nombre de mots communs entre deux libellés :

$$d_{norm}(A, B) = \frac{|A \cap B|}{\max(|A|, |B|)}$$

$$o\text{ù } A \text{ et } B \text{ sont des ensembles de mots}$$

Cette formule modifiée fournit une mesure de la similitude basée sur la proportion de mots communs, ce qui est plus pertinent pour les libellés détaillés. Elle est moins sensible à l'ordre exact des mots et à la longueur globale des chaînes, se concentrant plutôt sur la présence et la fréquence de mots-clés communs.

En normalisant la distance de Jaccard de cette manière, nous obtenons une évaluation plus précise et équilibrée de la correspondance entre les libellés complexes des véhicules, ce qui est crucial pour notre processus de réconciliation des données.

4.2.7 Distance générale

Afin d'évaluer de manière globale la correspondance entre chaque véhicule du référentiel et le véhicule à tester, nous procédons au calcul de la distance générale. Cette distance est cruciale pour déterminer le degré de similitude globale entre les véhicules.

La distance générale est obtenue par la sommation pondérée des distances individuelles calculées pour chaque dimension. Ainsi pour calculer la distance générale entre un véhicule test et un véhicule issu du référentiel, la méthode se formalise avec la formule suivante :

$$\forall veh_{ref} => d(veh_{test}, veh_{ref}) = \sum_{i=1}^{n} p_i * d_i$$

$$\text{où } p_i \textit{ est la pondération de la } i_{\text{ème}} \textit{ dimension du véhicule } veh_{ref}$$
$$\textit{et } d_i \textit{ la distance calculée de la } i_{\text{ème}} \textit{ dimension du véhicule } veh_{ref}$$

La formule est à appliquer à tous les véhicules candidats issus du référentiel, ce qui permettra par la suite de faire un tri du plus proche au plus éloigné.

4.2.8 Les transcodifications

Dans notre démarche de réconciliation, un élément essentiel est la correction et la standardisation des données, particulièrement pour les dimensions critiques telles que la marque, le modèle et l'énergie des véhicules. Pour traiter efficacement les incohérences et les erreurs courantes dans ces dimensions, nous utilisons le processus de transcodification.

La transcodification est une technique de mise en qualité des données qui implique l'utilisation de paires clé-valeur pour corriger et standardiser des informations. Dans ce contexte, chaque clé représente une version mal orthographiée ou incorrecte d'une marque ou d'un modèle de véhicule (par exemple, une faute de frappe ou une variation non standard), tandis que la valeur associée est la forme correcte et standardisée de cette marque ou modèle.

Par exemple, une transcodification peut corriger "Renolt" en "Renault" ou "BMV" en "BMW". Cette approche est particulièrement utile pour gérer les variations et les erreurs fréquentes dans les données provenant des DMS, où des fautes de frappe ou des abréviations peuvent entraîner des incohérences.

Le processus de transcodification se déroule en deux étapes principales :

- **Identification des Variations** : Nous analysons d'abord les données pour identifier les variations et les erreurs courantes dans les dimensions clés. Cette analyse peut être réalisée en examinant les données historiques ou en utilisant des techniques d'analyse textuelle pour détecter des patterns communs d'erreurs.
- **Application des Transcodifications** : Une fois les variations identifiées, nous appliquons les transcodifications pour corriger les données. Chaque instance incorrecte est remplacée par sa forme standardisée, en se basant sur notre base de données de paires clé-valeur.

Ce processus garantit que les dimensions critiques comme la marque, le modèle et l'énergie sont uniformisées et précises avant d'entrer dans le processus de réconciliation. Cela améliore non seulement la qualité des données, mais augmente également la précision du rapprochement en éliminant les variations qui pourraient fausser les calculs de distance.

En intégrant les transcodifications dans notre processus de réconciliation, nous parvenons à une plus grande fiabilité des correspondances et à une meilleure qualité globale des résultats de réconciliation.

4.2.9 Les Soundex, Phonex et Double Métaphone

Dans notre processus de mise en qualité des données pour la réconciliation, nous intégrons également l'utilisation des algorithmes Soundex, Phonex et Double Métaphone. Bien qu'ils ne soient pas directement utilisés dans le processus de réconciliation lui-même, ces algorithmes jouent un rôle complémentaire en fournissant des indications lors de la transcodification automatique.

Soundex et Phonex, conçus principalement pour l'anglais, ainsi que Double Métaphone, plus adapté aux variations linguistiques, transforment les mots en un code représentant leur sonorité. Cette fonctionnalité est particulièrement utile pour identifier les mots qui sonnent de manière similaire mais orthographiés différemment. Cela aide à détecter les erreurs potentielles et les variations dans les noms de marque ou modèle.

Cependant, il est important de souligner que Soundex et Phonex ont certaines limitations, notamment leur conception principalement pour la langue anglaise, ce qui peut réduire leur efficacité et fiabilité lorsqu'ils sont appliqués à des données en français ou dans d'autres langues. En raison de cette limitation, l'utilisation de ces algorithmes dans notre processus est principalement informative et ne remplace pas les méthodes plus robustes de transcodification et de correction des données.

Double Métaphone, quant à lui, offre une meilleure adaptabilité linguistique. Mais reste d'une fiabilité relativement limitée.

En résumé, bien que Soundex, Phonex et Double Métaphone ne soient pas des outils fiables pour une utilisation directe dans la réconciliation des données véhicules, leur contribution en tant qu'outils d'aide à la transcodification est précieuse. Ils offrent une perspective supplémentaire dans l'identification des variations et des erreurs potentielles dans les données, mais leur utilisation reste limitée à un rôle informatif en raison de leurs contraintes inhérentes, et doivent nécessairement être validées ensuite par un expert métier.

4.2.10 Les méthodes de nettoyage de données

Afin de permettre la mise en pratique de tous ces concepts de façon cohérentes, la plupart des données devront être préalablement nettoyées.

Pour ce faire, nous emploierons quelques méthodes suivant les cas :

- Suppression des caractères spéciaux et accentués
- Suppression des effets dus à la casse en mettant toutes les chaines en majuscules
- Suppression des espaces dans les chaines de caractères

Ces méthodes permettront dans la plupart des cas de limiter voire de supprimer les effets de fautes de frappes. En permettant la comparaison entre des chaines standardisées suivant les mêmes règles.

4.2.11 Synthèse des concepts théoriques

En conclusion de cette section consacrée aux concepts théoriques, il est essentiel de souligner la cohérence et la complémentarité des méthodes et des outils que nous avons explorés pour la réconciliation des données véhicules.

Tout d'abord, la segmentation des caractéristiques des véhicules en dimensions distinctes fournit une base structurée pour l'analyse. Cette approche permet une comparaison précise et ciblée entre les caractéristiques des véhicules et celles contenues dans le référentiel Argus, éliminant ainsi les ambiguïtés et optimisant la pertinence des correspondances.

Les différentes méthodes de calcul de distance que nous avons examinées (discrète, euclidienne, de Levenshtein, et de Jaccard) sont spécialement adaptées à la nature des données traitées, qu'elles soient numériques ou textuelles. La normalisation de ces distances garantit une évaluation équitable et proportionnelle des similitudes entre les dimensions des véhicules.

L'introduction de la pondération dans le calcul de la distance générale offre une flexibilité supplémentaire, permettant de hiérarchiser les dimensions selon leur importance relative dans le processus de réconciliation. Cette pondération s'avère cruciale pour affiner les résultats et les adapter aux besoins spécifiques des différents partenaires.

Les transcodifications apportent une valeur ajoutée significative en améliorant la qualité des données avant leur entrée dans le processus de réconciliation. Cette étape de prétraitement assure une plus grande précision dans les correspondances établies entre les véhicules.

Enfin, bien que les algorithmes Soundex, Phonex et Double Métaphone ne soient pas directement utilisés pour la réconciliation, leur rôle dans la suggestion de corrections potentielles pour la transcodification automatique renforce la robustesse de notre approche, malgré leur nature principalement informative.

Fort de ces fondements théoriques, nous nous apprêtons maintenant à aborder la prochaine phase cruciale de notre projet : la définition de l'architecture de l'application. Cette étape vise à traduire les concepts et les méthodes que nous avons développés en un cadre fonctionnel et pratique. L'architecture de l'application jouera un rôle essentiel en orchestrant l'interaction des différents composants et en définissant la manière dont les données seront traitées, analysées et présentées. Cette phase représente ainsi la concrétisation de notre démarche théorique et la préparation du terrain pour la mise en œuvre effective de notre solution de réconciliation.

4.3 Définition de l'architecture générale

4.3.1 Périmètre de l'application au sein du SI

Compte tenu de l'étendue et de la complexité du SI de CGI Finance, il était primordial de définir le périmètre de l'application dédiée à la réconciliation.

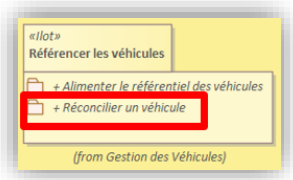


Figure 18 – Diagramme d'architecture fonctionnelle

Voir annexe 4.3.1

Ainsi, en collaboration avec l'architecte, il a été déterminé que l'application RMOD (Réconcilier un Véhicule) s'intégrerait au sein du diagramme d'architecture fonctionnelle, localisée précisément:

- Dans la zone « Financement Auto »
- Au cœur du quartier « Gestion des véhicules »
- Et ancré dans l'îlot « Référencer les véhicules »

Cette localisation est cohérente avec l'objectif de l'application, qui vise à améliorer la qualité des données et à réconcilier les flux de données de véhicules traversant le SI.

4.3.2 Liaison entre vue fonctionnelle et applicative

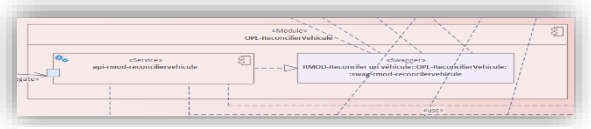


Figure 19 - Schéma de liaison entre architecture fonctionnelle et applicative

Voir Annexe 4.3.2

Le schéma en annexe 4.3.2 montre l'architecture applicative cible de la solution à développer, il fait le lien entre la vue fonctionnelle et la vue applicative, en relation avec le diagramme d'architecture fonctionnelle vu dans la partie précédente.

Nous détaillerons ces deux vues dans les deux parties prochaines.

4.3.3 Vue fonctionnelle

Le système sera donc, d'un point de vue fonctionnel, divisée en 4 services distincts, en corrélation avec l'implémentation du service Swagger d'un point de vue applicatif, qui exposera les services et endpoints de manière structurée :

- Service « Accéder aux Transcodifications » : regroupe les méthodes de création, consultation, de filtrage, ainsi que de validation et d'abandon des transcodifications.
- Service « Accéder aux Réconciliations Modèle Véhicule » : regroupe les méthodes de création et de consultation des réconciliations.
- Service « Accéder aux Stratégies de Réconciliation Modèle Véhicule » : regroupe les méthodes de création, consultation, ainsi que de validation et d'abandon des stratégies de réconciliation.

  Note : Les stratégies de réconciliations, nous le verrons par la suite, sont des objets qui contiendront le paramétrage nécéssaire (pondérations etc) pour effectuer une réconciliation.
- Service « Réconcilier un véhicule » : regroupe les méthodes permettant la mise en qualité (transcodification automatique et enrichissement des données), la réconciliation par caractéristique et la réconciliation par immatriculation.

Il est à noter également l'exposition de différents objets métiers qui correspondront aux objets principaux des différents services sur lesquels nous reviendront en détail dans la partie dédiée :

- Transcodification information véhicule
- Réconciliation Modèle Véhicule
- Stratégie Réconciliation Modèle Véhicule

4.3.4 Vue applicative

D'un point de vue applicatif, le système sera cette fois subdivisé en 3 modules distincts :

- Module « OPE-reconcilierVehicule » :

  Module opérationnel comprenant les services suivants :

  - « api-rmod-reconciliervehicule » : Composant cœur du système contenant l'implémentation de toutes les opérations métier autour des objets métiers du système. C'est un composant .NET.
  - « swag-rmod-reconciliervehicule » : Contrat de service du système. C'est un composant de type Swagger.

- Module « BDD-reconcilierVehicule » :

  Module gérant la base de données du système, ne comprenant que le service « bdd-rmod-reconciliervehicule ». Qui est un composant PostgreSQL.

- Module « BATCH-reconcilierVehicule » :

  Module comprenant les différents services Batch nécessaires au système :

  - « batch-rmod-majvehiculeimmatricule » : Batch de mise à jour de la table des véhicules immatriculés. C'est un composant ETL (avec DataStage).
  - « batch-rmod-majtranscodification » : Batch de mise à jour des transcodifications. Composant ETL également.
  - « batch-rmod-initvehiculeimmatricule » : Batch d'initialisation des véhicules immatriculés. Composant ETL également.

Il est à noter qu'apparaissent également sur le schéma (Annexe 4.3.2) les services appelants venant d'autres systèmes, ainsi que les services appelés, dont le service externe « Argus » qui correspond au SIV Argus.

En effet, dans le cas de la réconciliation par immatriculation, il a été décidé en cas d'échec, de continuer à appeler le SIV Argus pour que les clients aient toujours une réponse.

Petite précision :

Par souci de délimitation, il est à noter que le sujet de ce mémoire se limitera aux modules « OPE-ReconcilierVehicule » et « BDD-ReconcilierVehicule », car ce sont les seuls dont nous avons eu la charge exclusive. Mais il faut savoir que nous avons dû, bien entendu, communiquer avec les acteurs qui ont mis en place le module de batch afin de se coordonner et avoir un ensemble cohérent.

4.3.5 Vue technique

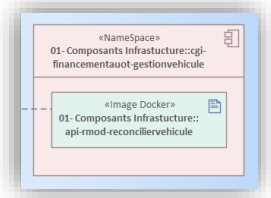


Figure 20 – Schéma de liaison entre architecture applicative et technique

Voir Annexe 4.3.5

Enfin, en considérant la vue technique, nous pouvons déterminer 3 environnements distincts, correspondant respectivement à aux 3 modules de la vue applicative, pour le déploiement et l'exécution du système.

Le module « OPE-ReconcilierVehicule » sera donc déployé via une image Docker sur la plateforme Kubernetes de l'entreprise.

Le module « BATCH-ReconcilierVehicule », quant à lui sera déployé via l'environnement DSX.

Et enfin, le module « BDD-ReconcilierVehicule » sera lui déployé dans la base de données PostgreSQL nommée CTB, et dans le schéma « rmod ».

4.3.6 Cas particulier de l'accès aux données S.I.V.

En ce qui concerne l'accès aux données véhicules du S.I.V., l'ajout un module distinct a été décidé, nommé « OPE-AccederInformationsANTS ».

Ce module prendra la forme donc d'une API distincte, interne, qui sera appelée par RMOD dans le cas de la réconciliation par immatriculation.

Ce module s'appuie également sur le module « BDD-reconcilierVehicule » en ce qui concerne la persistance des données. Données qui sont alimentées et mises à jour par le module « BATCH-reconcilierVehicule », en particulier les services « batch-rmod-initvehiculeimmatricule » et « batch-rmod-majvehiculeimmatricule », qui auront pour responsabilité de :

- Récupérer les fichiers csv sur le serveur du ministère.
- Intégrer les données non existantes dans le module « BDD-reconcilierVehicule ».
- De mettre à jour les données déjà présentes.
- De vérifier si les transcodifications existent pour les nouvelles données entrantes.
- D'ajouter dans la table des transcodifications du module « BDD-reconcilierVehicule » les identifiants attendus des transcodifications non existantes, afin de tenter une transcodification automatique de masse ensuite.

Nous n'entrerons pas trop dans les détails du fonctionnement de tout ceci, car étant principalement de la responsabilité et nous n'en avons pas eu la charge directe (mis à part la transcodification automatique et le développement de l'API d'accès). Néanmoins, il est nécessaire d'avoir une idée de ces mécanismes pour une compréhension globale.

4.3.7 Conclusion sur l'architecture générale

L'architecture de l'application RMOD, telle qu'elle a été structurée et définie, représente un jalon crucial pour le projet.

Chaque module, conçu minutieusement, s'intègre harmonieusement dans l'infrastructure complexe de CGI Finance, tout en offrant la flexibilité et la capacité d'évolution nécessaires pour s'adapter aux exigences métier changeantes.

La définition claire du périmètre de l'application, ainsi que la distinction bien établie entre les vues fonctionnelle, applicative et technique, constituent une base solide pour la suite de notre travail. Avec l'intégration du module spécifique « OPE-AccederInformationsANTS », dédié à l'accès aux données du S.I.V., notre approche se trouve renforcée, permettant une couverture complète des besoins de réconciliation.

Cette architecture globale nous prépare idéalement à entrer dans la phase suivante de notre mémoire, où nous explorerons en détail les aspects plus spécifiques de cette architecture.

4.4 Architecture détaillée

4.4.1 Les listes statiques

Les listes statiques sont des collections immuables de données qui servent de références constantes et fiables au sein de l'application. Elles sont utilisées pour stocker des informations qui ne nécessitent pas de mise à jour fréquente.

Elles seront utilisées par les différents objets du système afin de conférer un statut particulier à certaines données.

Elles sont au niveau implémentation déployées sous forme d'une bibliothèque partagée entre le système et ses clients.

4.4.1.a Liste Dimensions Véhicule

Liste qui énumère l'ensemble des dimensions vues précédemment qui seront utilisables à terme pour la réconciliation.

ENUM	Description
MARQUE	Marque du véhicule
MODELE	Modèle du véhicule
SOUSMODELE	Sous modèle du véhicule
GENERATION	Génération du véhicule
FINITION	Finition du véhicule
PHASE	Phase du véhicule
VERSION	Libellé long du véhicule
ENERGIE	Energie du véhicule
BOITE_DE_VITESSE	Boite de vitesse précise du véhicule (ex : BVA5)
TYPE_BOITE_VITESSE	Type de boite de vitesse du véhicule (ex : Manuelle)
NOMBRE_DE_PLACES	Nombre de places du véhicule
NOMBRE_DE_PORTES	Nombre de portes du véhicule
PUISSANCE_FISCALE	Puissance fiscale du véhicule
PUISSANCE_DIN	Puissance réelle du véhicule
EMISSION_CO2	Emission CO2 du véhicule
CYLINDREE	Cylindrée du véhicule
CONSOMMATION	Consommation du véhicule
POIDS	Poids du véhicule
PRIX_CATALOGUE	Prix neuf du véhicule
DATE_PREMIERE_MEC	Date de première mise en circulation du véhicule
CAROSSERIE	Carrosserie du véhicule (ex : Berline, Break)

4.4.1.b Liste Méthodes Transcription

Liste utilisée dans les transcodifications qui énumère le type de traitement utilisé pour la réaliser.

ENUM	Description
MANUELLE	Transcription de la valeur en entrée effectuée à partir d'une transcodification validée
AUTOMATIQUE	Transcription de la valeur en entrée effectuée à partir d'un traitement automatique
AUCUNE	Aucune transcription entre la valeur en entrée et la valeur en sortie
NON_TRANSFORMABLE	Détection d'une erreur de format

4.4.1.c Liste Détail Réconciliation Modèle Véhicule

Liste énumérant les différents statuts précis possible d'une réconciliation.

ENUM	Description
UN_SEUL_MODELE_AVANT_TRI	Un seul résultat correspondant identifié avant l'application des critères de tri.
UN_SEUL_MODELE_APRES_TRI	Un seul résultat correspondant identifié après l'application des critères de tri.
PLUSIEURS_MODELES_AVEC_DIFFERENCE_PRIX_INFERIEUR_A	Plusieurs modèles correspondants présentent un écart de prix inférieur à un seuil prédéfini, permettant une sélection flexible.
PLUSIEURS_MODELES_AVEC_DIFFERENCE_PRIX_SUPERIEUR_A	Plusieurs modèles correspondants présentent un écart de prix supérieur à un seuil prédéfini, nécessitant une analyse approfondie.
AUCUN_MODELE	Aucun résultat

Les critères de tri et de seuil cités dans ce tableau feront l'objet d'une explication lorsque nous aborderons la partie sur l'objet métier des stratégies de réconciliation.

4.4.1.d Liste Statut Réconciliation Modèle Véhicule

Liste énumérant les différents statuts généraux possible d'une réconciliation

ENUM	Description
SUCCES	Réconciliation considérée comme réussie.
ECHOUEE	Réconciliation considérée comme en échec.

En se basant sur les statuts précis de réconciliation vus au-dessus, nous pouvons, avec les consignes métier, établir une correspondance entre eux.

Liste Détails Réconciliation Modèle Véhicule	Liste Statut Réconciliation Modèle Véhicule
UN_SEUL_MODELE_AVANT_TRI	**SUCCES**
UN_SEUL_MODELE_APRES_TRI	
PLUSIEURS_MODELES_AVEC_DIFFERENCE_PRIX_INFERIEUR_A	
PLUSIEURS_MODELES_AVEC_DIFFERENCE_PRIX_SUPERIEUR_A	**ECHOUEE**
AUCUN_MODELE	

4.4.1.e Liste Statut Stratégie Réconciliation Modèle Véhicule

Liste énumérant les différents statuts du cycle de vie des stratégies de réconciliation.

ENUM	Description
INITIALISEE	La stratégie existe et est initialisée (non utilisable).
VALIDEE	La stratégie existe et est validée (utilisable).
ABANDONNEE	La stratégie existe et est abandonnée (non utilisable).

L'utilisation de ce statut sur les stratégie permettra de s'assurer que celle utilisée est bien acceptée par le système.

4.4.1.f Liste Statut Transcodification

Liste énumérant les différents statuts du cycle de vie des transcodifications.

ENUM	Description
INITIALISEE	La transcodification existe et est initialisée (non utilisable).
VALIDEE	La transcodification existe et est validée (utilisable).
ABANDONNEE	La transcodification existe et est abandonnée (non utilisable).

L'utilisation de ce statut sur les transcodifications permettra de s'assurer que celle utilisée est bien acceptée par le système.

De plus, l'utilisation de deux objets liste statique différents pour les stratégies et transcodification pourrait paraitre redondant, mais cela permet de respecter le principe de responsabilité unique compris dans les principes S.O.L.I.D. [12].

4.4.1.f Liste Statut Transcodification

Liste énumérant les différents types de traitement possibles dans la mise en qualité des données de véhicule.

ENUM	Description
ENRICHISSEMENT	A utiliser pour effectuer un enrichissement à partir du champ Version.
TRANSCODIFICATION	A utiliser pour effectuer une transcodification automatique.

Ces différents statuts seront à utiliser pour indiquer au système si l'on veut effectuer une transcodification automatique ou un enrichissement de données à partir du champ Version.

4.4.2 Les objets métier

Les objets métiers sont des conceptions logicielles qui modélisent des entités concrètes ou concepts propres à un domaine d'activité.

4.4.2.a Transcodification Information Véhicule

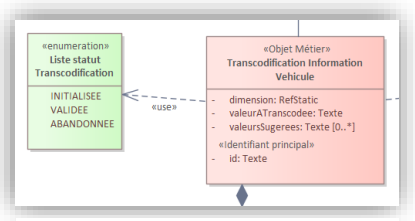


Figure 21 - Objet métier Transcodification Information Véhicule

Voire Annexe 4.4.2.a

Cet objet métier représente les informations de transcodification qui permettent de corriger et standardiser les données relatives aux véhicules.

Ci-dessous les différentes étapes du cycle de vie de l'objet, matérialisées par la liste statique « Liste Statut Transcodification », avec les différentes opérations nécessaires pour passer d'un statut à l'autre.

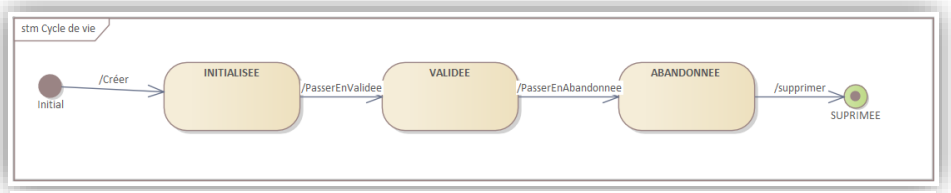


Figure 22 - Cycle de vie de l'objet métier Transcodification Information Véhicule

L'objet métier Transcodification Information Véhicule comprend un identifiant principal unique sous forme de texte qui devra répondre à ces règles de nommage suivant leur type :

- Transcodification MARQUE :

  « MARQUE » + « , » + « VALEUR INITIALE »

- Transcodification ENERGIE :

  « ENERGIE » + « , » + « VALEUR INITIALE »

- Transcodification BOITE DE VITESSE :

  « BOITE_DE_VITESSE » + « , » + « VALEUR INITIALE »

- Transcodification MODELE :

  « MODELE » + « , » + « VALEUR MARQUE TRANSCODEE » + « , » + « VALEUR INITIALE »

  Par conséquent, il sera toujours nécessaire d'avoir la marque transcodée pour avoir une transcodification modèle.

De plus, un sous objet « Validation » contenant la valeur transcodée sera présent dans le cas où cette valeur existe.

Et également un tableau « valeurSuggerees » qui sera garni par la transcodification automatique.

4.4.2.b Stratégie Réconciliation Modèle

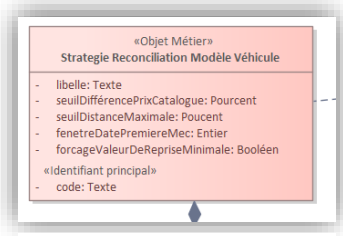


Figure 23 - Objet métier Stratégie Réconciliation Modèle

Voir Annexe 4.4.2.b

Cet objet métier contient les paramètres de configuration, tels que les pondérations et autres critères, nécessaires pour adapter le processus de réconciliation aux spécificités des données en entrée.

Ci-dessous le cycle de vie de l'objet métier, très similaire à celui des transcodifications, mais en se basant sur la liste statique « Liste Statut Stratégie Réconciliation Modèle Véhicule ».

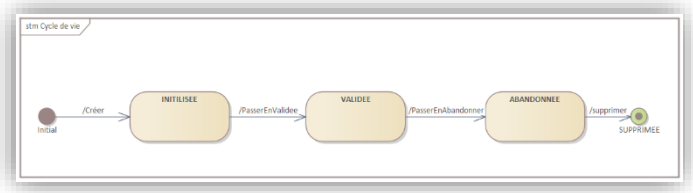


Figure 24 - Cycle de vie de l'objet Stratégie Réconciliation Modèle

L'objet métier en lui-même comprend plusieurs caractéristiques :

- Un identifiant principal unique en texte qui doit être choisi par l'utilisateur.
- Une propriété « libellé » choisie par l'utilisateur qui comprendra une description de la stratégie en question.
- Une propriété seuilDifférencePrixCatalogue qui est celle déjà évoquée plus haut. Elle représente un pourcentage de différence de prix catalogue entre le candidat le plus cher et le moins cher maximum pour considérer la réconciliation comme réussie ou non dans le cas de plusieurs véhicules possibles.
- Une propriété seuilDistanceMaximale, exprimée en pourcentage de la distance minimale des candidats. Elle permet de sélectionner les candidats qui ont une distance limite relative à la distance minimale calculée de tous les candidats.
- Une propriété fenetreDatePremiereMec exprimée par un entier, correspondant au nombre de mois que l'on accepte dans la sélection des véhicules candidats après la date de mise en circulation. En effet, dans le catalogue constructeur, nous n'avons que les dates de commercialisation, or un véhicule peut (et c'est souvent le cas) ne pas être vendu de suite. Il faut donc déterminer un intervalle de date (déterminé par analyse) dans lesquelles les véhicules peuvent être sélectionnés.
- Une propriété ForcageValeurDeRepriseMinimale qui forcera le système à ne sélectionner que le candidat qui a le prix catalogue le plus bas.
- Une relation dimensionsPonderees qui portera la liste des pondérations, associées au type de dimension dans lesquelles elles devront être utilisées.

4.4.2.c Réconciliation Modèle Véhicule

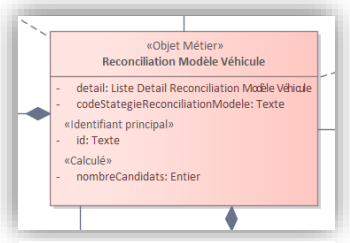


Figure 25 - Objet métier Réconciliation Modèle Véhicule

Voir Annexe 4.4.2.c

Enfin, le troisième et dernier objet métier du système contient le résultat de la réconciliation pour un véhicule donné.

Il regroupe toutes les informations qui ont servi à une réconciliation en particulier, comme les « infosVehicule » d'origine, ainsi le code de la stratégie utilisée.

Il porte également les différents statuts de réconciliation dans les deux listes statiques « Liste Détail Réconciliation Modèle Véhicule » et « Liste Statut Réconciliation Modèle Véhicule ».

Et les résultats en tant que tel sont représentés par une liste « candidats », dans laquelle se trouve la distance générale pour chaque candidat, l'id argus associé, ainsi que la liste des dimensions utilisées avec, pour chacune d'entre elles, son type et sa distance (hors pondération).

Enfin, dans le cas d'un résultat certain, un objet « candidatRetenu » est également présent avec sa distance générale et son id argus.

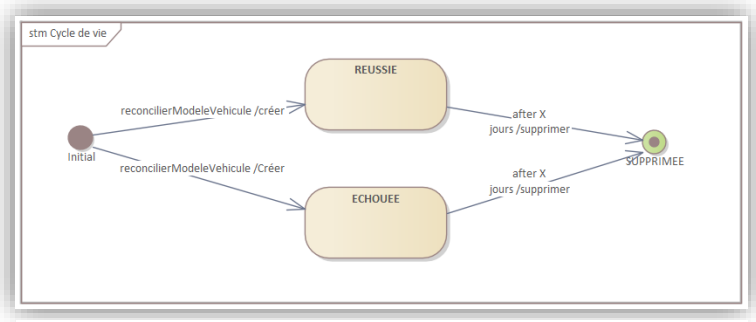


Figure 26 - Cycle de vie de l'objet Réconciliation Modèle Véhicule

Le cycle de vie de cet objet se base sur la liste statique « Liste Statut Réconciliation Modèle Véhicule », à laquelle est attribué une valeur à la création. Et après un certain nombre de jours, l'objet en base est supprimé.

4.4.2.d Conclusion sur les objets métiers

Le système entier repose donc sur ces trois objets métiers qui seront au cœur de l'intégralité du processus de réconciliation.

Ce seront en outre les objets qui seront rendu persistants dans le module « BDD-ReconcilierVehicule », afin de les rendre disponibles à tout moment, d'effectuer un suivi ainsi qu'une historisation en cas de besoin.

Ces objets seront englobés dans différents objets « échange » par la suite, qui correspondront aux objets d'entrée (les objets « demandes ») et de sortie (les objets « réponses ») de l'API, ce que nous aborderons dans la prochaine partie qui s'attardera de façon plus spécifique sur l'architecture des différents endpoints.

4.4.3 Les fonctionnalités du service « Accéder aux Transcodifications »

Ce service regroupe les différentes opérations C.R.U.D. en relation avec la gestion des transcodifications. Chaque fonctionnalité, et ceci sera valable pour chaque service, sera représenté par un endpoint API.

4.4.3.a Fonctionnalité Consulter

Cette fonctionnalité permet de consulter une transcodification stockée dans le système. Elle prend dans l'URN l'id de la transcodification et donne en sortie un objet « Résultat Consultation Transcodification Information Véhicule ».

URN : /transcodifications/{idtranscodification}

Verbe HTTP : GET

Objet résultat : Résultat Consultation Transcodification Véhicule

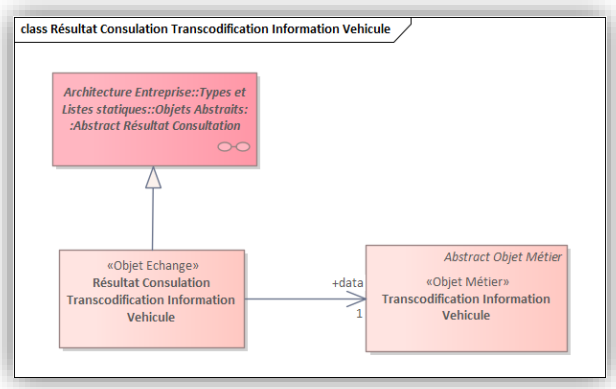


Figure 27 - Objet Résultat Consultation Transcodification Véhicule

Diagramme de séquence :

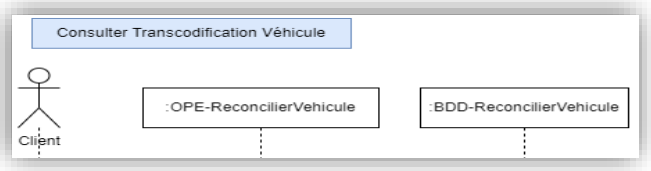


Figure 28 - Diagramme de séquence Consulter Transcodification Véhicule

Voir Annexe 4.4.3.a

4.4.3.b Fonctionnalité Créer

Cette fonctionnalité permet d'enregistrer une « Transcodification Information Véhicule » dans le système. Elle prend en entrée un objet d'échange « Demande Création Transcodification Information Véhicule » et renvoie en sortie un objet d'échange « Résultat Création Transcodification Information Véhicule ».

URN: /transcodifications

Verbe HTTP: POST

Objet demande : Demande Création Transcodification Information Véhicule

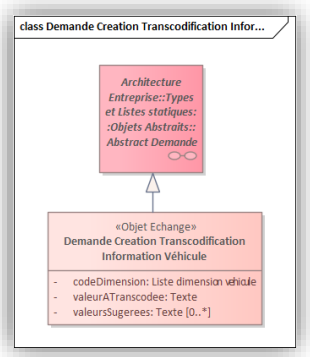


Figure 29 - Objet Demande Création Transcodification Information Véhicule

Objet résultat : Résultat Consultation Transcodification Véhicule

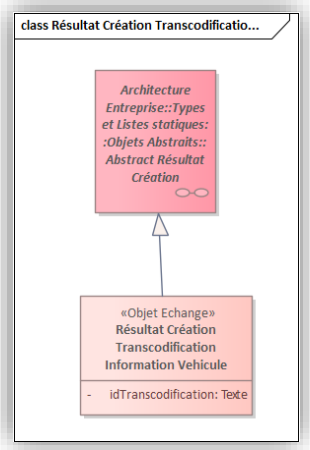


Figure 30 - Objet Résultat Transcodification Information Véhicule

Diagramme de séquence :

Voir Annexe 4.4.3.b

4.4.3.c Fonctionnalité Passer en Validée

Cette fonctionnalité permet de passer au statut validée une transcodification stockée dans le système. Elle prend dans l'URN l'id de la transcodification, avec un objet « Demande Passage Transcodification En Validée » dans le body et donne en sortie un code HTTP représentatif de l'état de l'opération.

De plus, cette fonctionnalité permet également la vérification de la valeur validée dans le cas de transcodification de type « MODELE ». En effet, c'est à ce stade qu'il va vérifier si la valeur de la propriété correspond à un modèle ou un sous-modèle. Dans ce deuxième cas, la validation transformera le type de la transcodification en SOUSMODELE.

URN : /transcodifications/{idtranscodification}/passerEnValidee

Verbe HTTP : POST

Objet demande : Demande Passage Transcodification En Validée

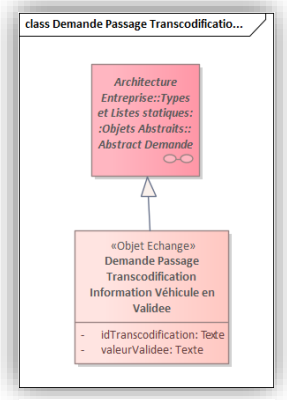


Figure 31 - Objet Demande Passage Transcodification En Validée

Codes résultat :

- 200 : opération réussie
- 400 : requête invalide (transcodification n'existe pas etc.)
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.3.c

4.4.3.d Fonctionnalité Passer en Abandonnée

Cette fonctionnalité permet de passer au statut abandonnée une transcodification stockée dans le système. Elle prend dans l'URN l'id de la transcodification, avec un objet « Demande Passage Transcodification En Validée » dans le body et donne en sortie un code HTTP représentatif de l'état de l'opération.

URN : /transcodifications/{idtranscodification}/passerEnAbandonnee

Verbe HTTP : POST

Objet demande : Demande Passage Transcodification En Abandonnée

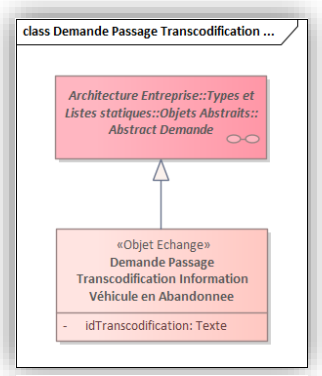


Figure 32 - Objet Demande Passage Transcodification En Abandonnée

Codes résultat :

- 200 : opération réussie
- 400 : requête invalide (transcodification n'existe pas etc.)
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.3.d

4.4.4 Les fonctionnalités du service « Accéder aux Stratégies de Réconciliation Modèle Véhicule »

Ce service regroupe les différentes opérations C.R.U.D. en relation avec la gestion des stratégies de réconciliation.

4.4.4.a Fonctionnalité Consulter

Cette fonctionnalité permet de consulter une stratégie de réconciliation stockée dans le système. Elle prend dans l'URN l'id de la stratégie et donne en sortie un objet « Résultat Consultation Stratégie Réconciliation Modèle Véhicule ».

URN : /strategiesReconciliationModeleVehicule/{codeStrategie}

Verbe HTTP : GET

Objet résultat : Résultat Consultation Stratégie Réconciliation Modèle Véhicule

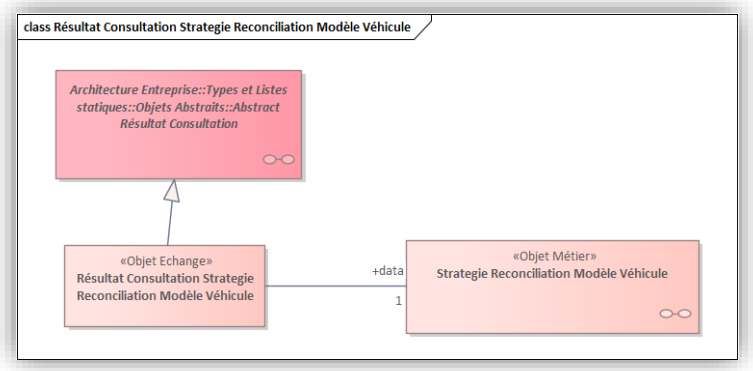


Figure 33 - Objet Résultat Consultation Stratégie Réconciliation Modèle Véhicule

Diagramme de séquence :

Voir Annexe 4.4.4.a

4.4.4.b Fonctionnalité Créer

Cette fonctionnalité permet d'enregistrer un objet « Demande Création Stratégie Réconciliation Modèle Véhicule » dans le système. Elle prend en entrée un objet d'échange « Demande Création Stratégie Réconciliation Modèle Véhicule » et renvoie en sortie code HTTP représentatif de l'état de l'opération.

URN : /strategiesReconciliationModeleVehicule

Verbe HTTP : POST

Objet demande : Demande Création Stratégie Réconciliation Modèle Véhicule

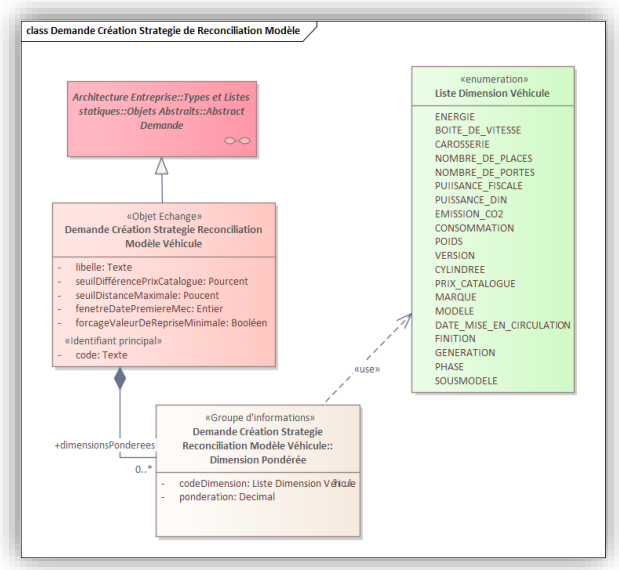


Figure 34 - Objet Demande Création Stratégie Réconciliation Modèle Véhicule

Codes résultat :

- 201 : opération réussie
- 400 : requête invalide (stratégie n'existe pas etc.)
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.4.b

4.4.4.c Fonctionnalité Passer en Validée

Cette fonctionnalité permet de passer au statut validée une stratégie stockée dans le système. Elle prend dans l'URN l'id de la stratégie, avec un objet « Demande Passage Stratégie Réconciliation En Validée » dans le body et donne en sortie un code HTTP représentatif de l'état de l'opération.

URN : /transcodifications/{idtranscodification}/passerEnValidee

Verbe HTTP : POST

Objet demande : Demande Passage Transcodification En Validée

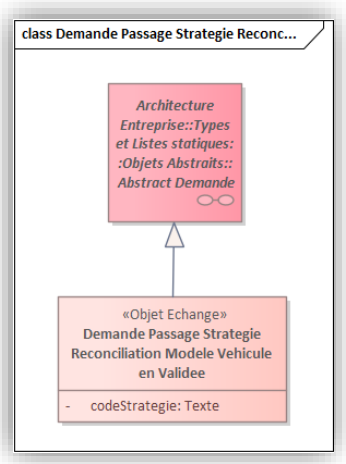


Figure 35 - Objet Demande Passage Stratégie Réconciliation Modèle Véhicule en Validée

Codes résultat :

- 200 : opération réussie
- 400 : requête invalide (transcodification n'existe pas etc.)
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.4.c

4.4.4.d Fonctionnalité Passer en Abandonnée

Cette fonctionnalité permet de passer au statut abandonnée une stratégie de réconciliation stockée dans le système. Elle prend dans l'URN l'id de la transcodification, avec un objet « Demande Passage Stratégie Réconciliation Modèle Véhicule En Abandonnée » dans le body et donne en sortie un code HTTP représentatif de l'état de l'opération.

URN : /transcodifications/{idtranscodification}/passerEnAbandonnee

Verbe HTTP : POST

Objet demande : Demande Passage Transcodification En Abandonnée

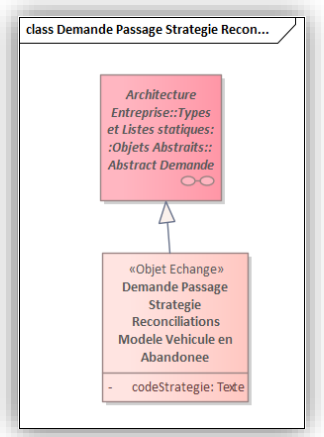


Figure 36 - Objet Demande Passage Stratégie Réconciliation Modèle Véhicule En Abandonnée

Codes résultat :

- 200 : opération réussie
- 400 : requête invalide (transcodification n'existe pas etc.)
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.4.d

4.4.5 Les fonctionnalités du service « Accéder aux Réconciliations Modèle Véhicule »

Ce service regroupe les fonctionnalités de création et consultation des réconciliation.

4.4.5.a Fonctionnalité Créer

Cette fonctionnalité permet de créer dans le système une entrée de l'objet métier Réconciliation Modèle Véhicule. Attention, elle n'effectue pas la réconciliation ! Elle prend dans le body un objet « Demande Création Réconciliation Modèle Véhicule » et renvoie un objet « Résultat Création Réconciliation Modèle Véhicule ». L'idReconciliation est automatiquement généré par le système à ce stade.

URN : /reconciliations

Verbe HTTP : POST

Objet demande : Demande Création Réconciliation Modèle Véhicule

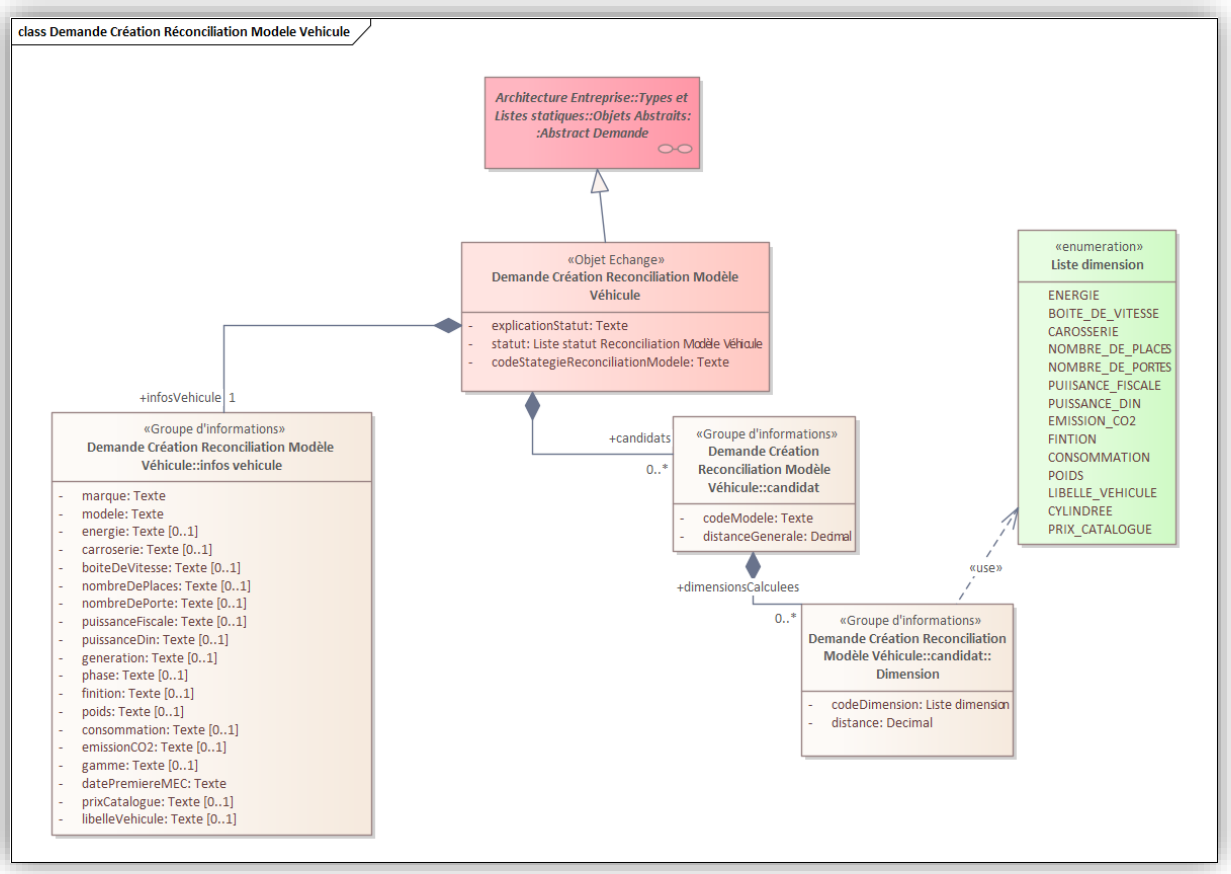


Figure 37 - Objet Demande Création Réconciliation Modèle Véhicule

Objet résultat : Résultat Création Réconciliation Modèle Véhicule

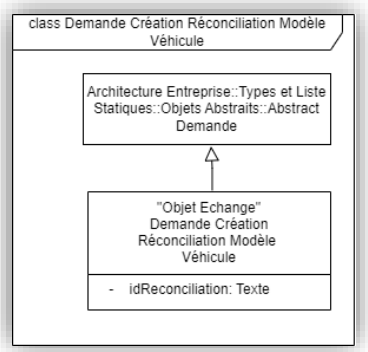


Figure 38 - Objet Résultat Création Réconciliation Modèle Véhicule

Codes résultat :

- 201 : opération réussie
- 400 : requête invalide (réconciliation existe déjà etc.)
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.5.a

4.4.5.b Fonctionnalité Consulter

Cette fonctionnalité permet de consulter une réconciliation stockée dans le système. Elle prend dans l'URN l'id de la réconciliation et donne en sortie un objet « Résultat Consultation Réconciliation Modèle Véhicule ».

URN : /reconciliations/{idReconciliation}

Verbe HTTP : GET

Objet résultat : Résultat Consultation Réconciliation Modèle Véhicule

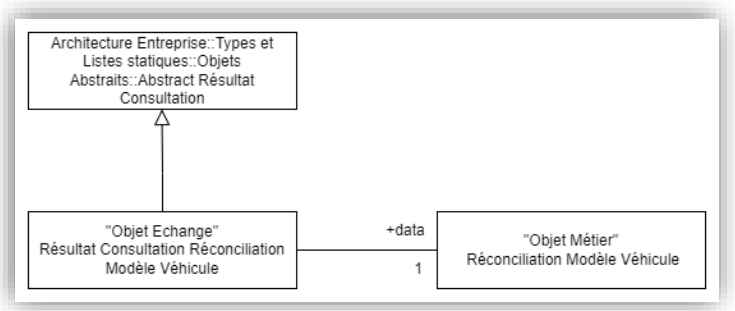


Figure 39 - Objet Résultat Consultation Réconciliation Modèle Véhicule

Codes résultat :

- 200 : opération réussie
- 404 : requête invalide (réconciliation n'existe pas etc.)
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.5.b

4.4.6 Les fonctionnalités du service « Réconcilier un Véhicule »

Ce service est le cœur du système, avec les fonctionnalités qui vont permettre la réconciliation de véhicule et la mise en qualité des données.

4.4.6.a Fonctionnalité Retranscrire

Cette fonctionnalité permet les opérations automatiques de mise en qualité des données de véhicule. En effet, suivant les paramètres passés, elle effectuera les opérations :

- De transcodification MODELE ou SOUSMODELE automatique
- D'enrichissement des dimensions éligibles à partir de la dimension VERSION

Suivant les cas et les règles métier définies, elle validera automatiquement les transcodifications ainsi définies, ou alors elle proposera une liste de suggestions, dans ce cas, une intervention humaine sera nécessaire pour la validation de la transcodification.

URN : /retranscrireInformationsVehicule

Verbe HTTP : POST

Objet demande : Demande Retranscription Informations Véhicule

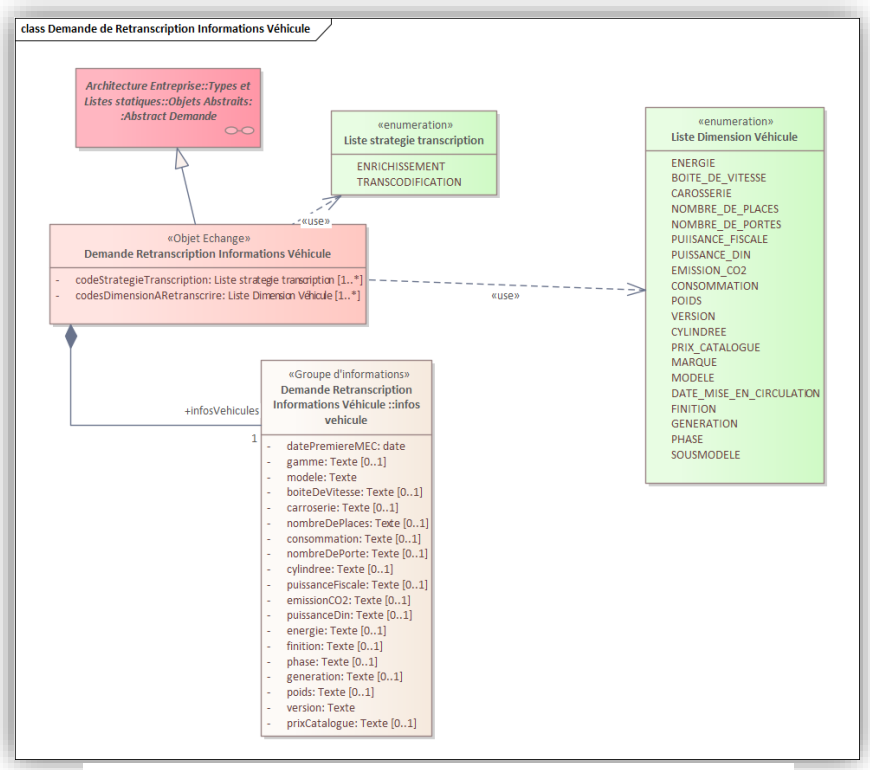


Figure 40 - Objet Demande Retranscription Informations Véhicule

Objet résultat : Résultat Retranscription Informations Véhicule

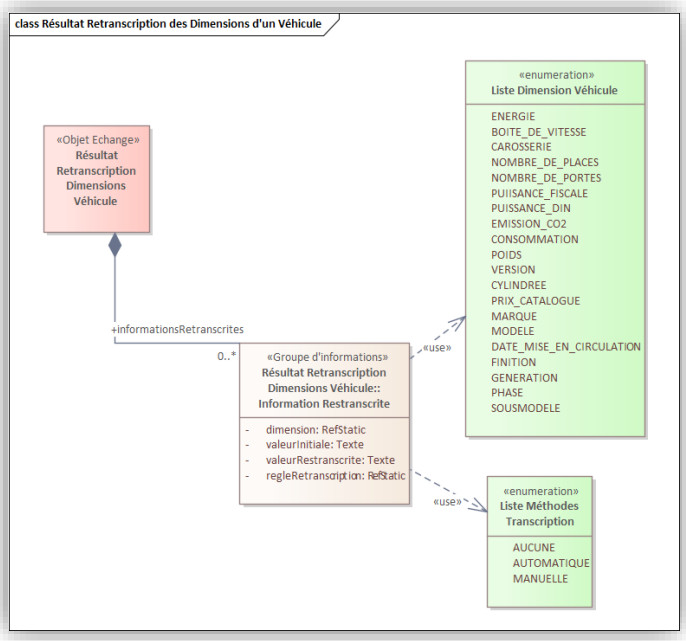


Figure 41 - Objet Résultat Retranscription Informations Véhicule

Codes résultat :

- 200 : opération réussie
- 400 : requête invalide (impossibilité d'obtenir un résultat)
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.6.a

Règles métier :

Il a été mis en place différentes règles de gestion afin de déterminer si la transcodification automatique ou l'enrichissement est un succès ou non.

Ces règles, assez restrictives pour éviter au maximum les effets de bords sur un système qui se veut automatisé, sont les suivantes :

- Transcodification :
  - Nettoyage des données (casse, caractères spéciaux et espaces)
  - Vérification si une et une seule donnée candidate correspond à celle en entrée :
    - Egalité parfaite
    - La donnée candidate est comprise dans celle en entrée (exemple : « RENAULT » est compris dans « RENAULTD »)
  - Si plusieurs ou aucun résultats, pas de validation automatique :
    - Ajout des résultats aux valeurs suggérées
    - Ajout des résultats des Soundex et Phonex aux valeurs suggérées
- Enrichissement :
  - Le champ VERSION doit exister
  - Nettoyage des données
  - Pour les types textuels (FINITION, MODELE)
    - Récupération des valeurs possibles du type de donnée à enrichir
    - Vérification si une de ces valeur est comprise dans le champ VERSION
  - Pour les types numériques :
    - Recherche d'un pattern spécifique dans le champ VERSION
    - Extraction de la donnée numérique si le pattern cherché existe

4.4.6.b Fonctionnalité Réconcilier par caractéristiques

Cette fonctionnalité, la principale, permet la réconciliation d'un véhicule à partir de ses caractéristiques avec des candidats du référentiel Argus (id argus).

Elle a besoin de quatre paramètres obligatoires :

- Marque après mise en qualité
- Modèle ou sous-modèle après mise en qualité
- Date première mise en circulation (date MEC)
- Energie après mise en qualité

De plus, un code de stratégie de réconciliation valide est également demandé pour l'appel, afin de l'appliquer pour la réconciliation. Si le code n'est pas valide, une stratégie par défaut est appliquée.

URN : /reconcilierModeleVehicule

Verbe HTTP : POST

Objet demande : Demande Réconciliation Modèle Véhicule

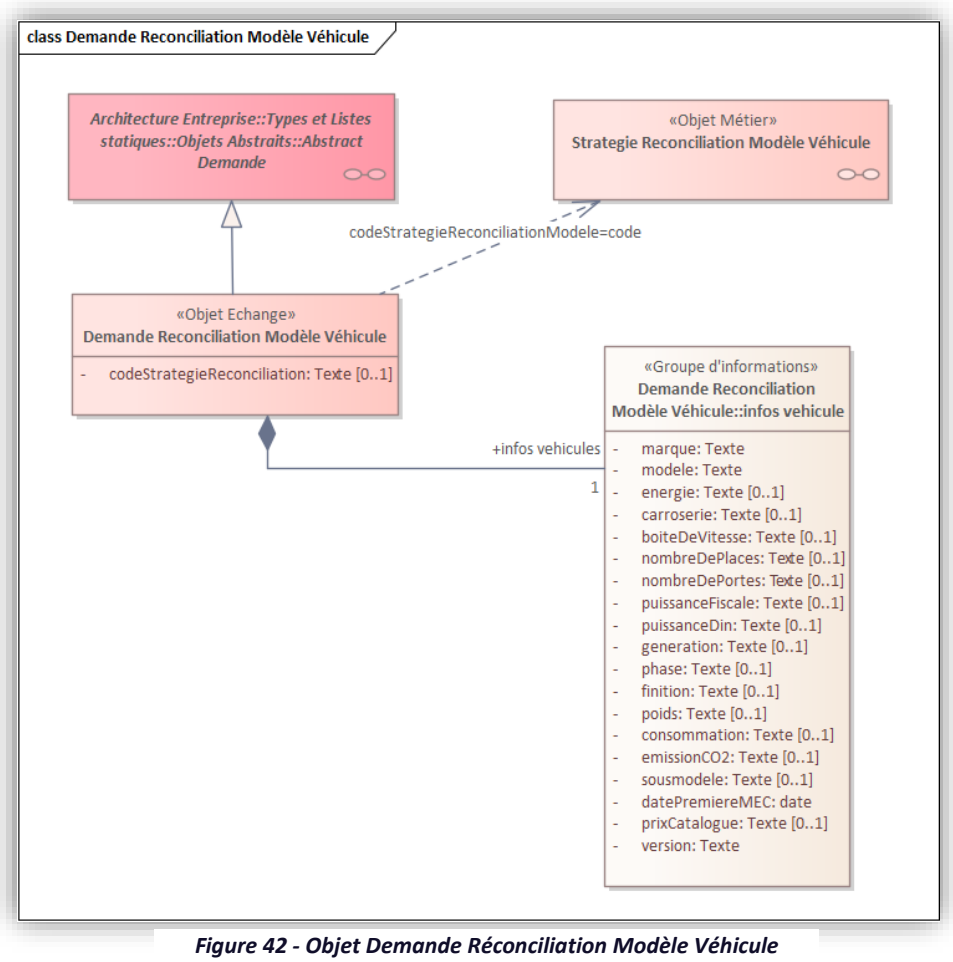


Figure 42 - Objet Demande Réconciliation Modèle Véhicule

Objet résultat : Résultat Réconciliation Modèle Véhicule

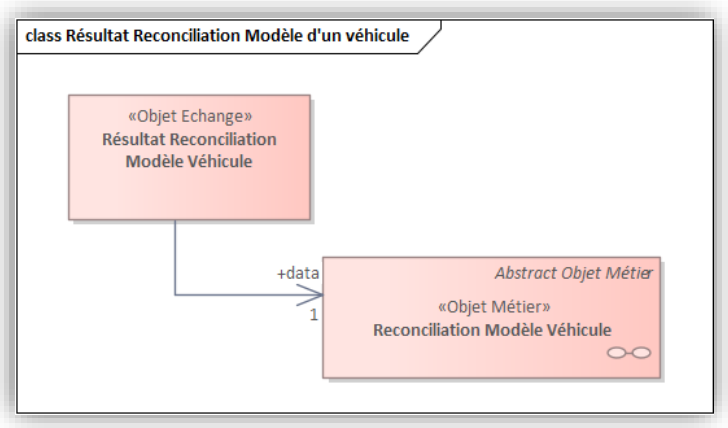


Figure 43 - Objet Résultat Réconciliation Modèle Véhicule

Codes résultat :

- 200 : opération réussie
- 400 : requête invalide (impossibilité d'obtenir un résultat)
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.6.b

Le diagramme de séquence représente certaines des opérations les plus importantes de la réconciliation, même s'il existera d'autres étapes que l'on abordera dans la partie développement.

Mais ici, nous constatons la présence des principales étapes dont :

- La récupération de la stratégie à partir du code (ou celle par défaut) et en utilisant la fonctionnalité déjà vue plus haut
- La transformation en dimensions du véhicule à tester, ainsi que les véhicules récupéré du référentiel (via les méthodes « Reduce »)
- Le calcul des distances
- L'application de différents filtres
- La vérification de l'énergie
- Le tri des véhicules candidat par distance

4.4.6.c Fonctionnalité Réconcilier par immatriculation

Enfin, la dernière fonctionnalité du système principal est la réconciliation par immatriculation. Qui pour une immatriculation donnée doit renvoyer la liste des id argus les plus probables.

Il est à noter qu'il existera deux endpoints pour cette fonctionnalité, un pour assurer la rétrocompatibilité avec l'outil existant, que je ne détaillerai pas ici car dès la conception il est considéré comme déprécié, et l'autre qui sera utilisée plus tard lorsque les clients pourront évoluer afin de consommer ce endpoint.

URN : /reconcilierModeleImmatriculation

Verbe HTTP : POST

Objet demande : Demande Réconciliation Modèle Immatriculé

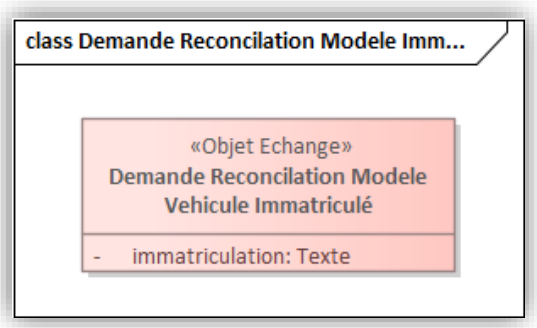


Figure 44 - Objet Demande Réconciliation Modèle Immatriculé

Objet résultat : Résultat Réconciliation Modèle Immatriculation

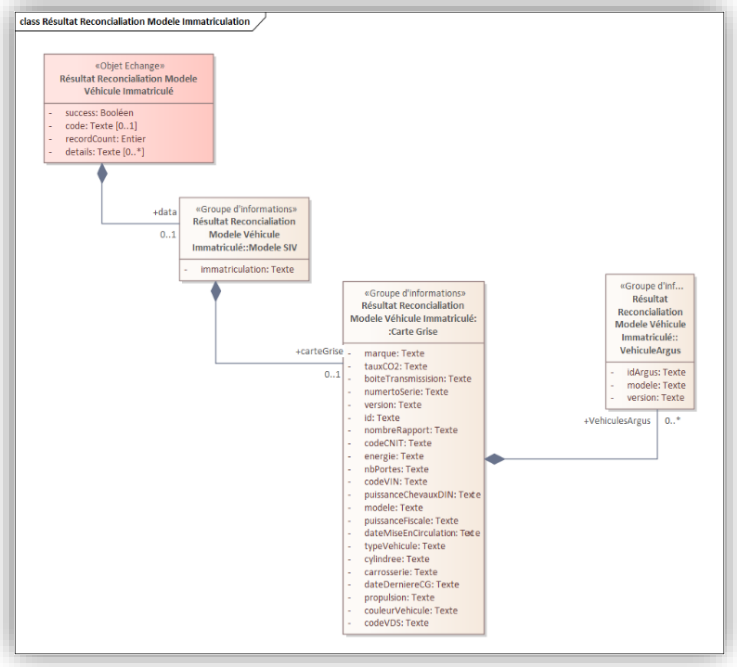


Figure 45 - Objet Résultat Réconciliation Modèle Immatriculation

Codes résultat :

- 200 : opération réussie
- 400 : requête invalide
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.6.b

Dans ce diagramme, nous pouvons remarquer l'apparition du service externe « Argus », qui n'est autre que le SIV Argus qui, comme dit plus haut, est censé être remplacé par ce système. Néanmoins, il a été décidé de le garder dans le cas où la réconciliation ne donnerait pas de résultat, pour assurer un résultat dans tous les cas (sauf si lui-même n'en a pas).

De plus, nous notons également l'apparition du service Informations Véhicule ANTS, qui sera détaillé dans la partie suivante.

4.4.7 Les fonctionnalités du service « Accéder aux Informations Véhicule ANTS »

Ce service a pour but de fournir les informations de carte grise d'un véhicule à partir de son immatriculation.

S'appuyant sur la base de données du SIV, il ne comporte qu'une seule fonctionnalité.

Il prend en entrée une immatriculation, et renvoie une liste d'informations carte grise du véhicule demandé.

En effet, comme vu précédemment, la subtilité ici est qu'une seule plaque d'immatriculation peut donner plusieurs résultats, et ceci suivant deux cas distincts :

- L'immatriculation est provisoire (commençant par « WW- »). Et dans ce cas, le véhicule à considérer est le plus récent.
- L'immatriculation est définitive. Malgré qu'une immatriculation définitive ne correspond bel et bien qu'à un seul véhicule, il peut y avoir plusieurs résultats car dans la base il y a une entrée par date de nouvelle carte grise (et donc de propriétaire ou de modification du véhicule). Donc, dans ce cas, pour la réconciliation, il faut prendre les données les plus anciennes associées à l'immatriculation.

Ce service ne fait pas la discrimination mais envoie toutes les données liées à une plaque. A la charge du client de choisir ensuite suivant ses critères. Et dans le cadre de la réconciliation par immatriculation vue juste au-dessus, c'est la fonction « ChoiceDataCarteGrise() » (cf. diagramme de séquence) qui gère ce choix.

4.4.8 Les services « BATCH » gérés par ETL

Malgré qu'ils soient au-delà de notre périmètre, nous allons décrire succinctement le rôle des services BATCH du module « BATCH-ReconcilierVehicule ».

Le « batch-rmod-initvehiculeimmatricule » va récupérer le fichier csv sur le serveur du ministère, décrypter les fichiers, et alimenter les données brutes dans la table « vehicules » du module « BDD-reconcilierVehicule » .

Le « batch-rmod-majtranscodification », lui, va vérifier dans les données nouvellement importées si les transcodification MARQUE, MODELE et ENERGIE existent. Si c'est le cas, il va alimenter les colonnes correspondant aux informations transcodées de la table « vehicules », et si ce n'est pas le cas, il va inscrire dans la table des transcodifications l'id que devrait avoir les transcodifications manquantes. Et ensuite appeler un nouveau service faisant partie de RMOD (« transcos-batch »), qui va chercher ces id, et appeler sur chacun d'entre eux la transcodification automatique.

Et enfin, le « batch-rmod-majvehiculeimmatricule » servira quant à lui à mettre à jour les données existantes dans la table « vehicules » en fonction des transcodifications existantes et de leur éventuelles modifications.

4.4.9 Le service « Transcos-BATCH »

Le service « Transcos-BATCH » est, comme vu précédemment, un service qui va automatiser les transcodifications automatiques sur les données du SIV importées par le module « batch-rmod-majtranscodification ».

Il est déclenché par un appel API via l'ETL qui effectue le batch de mise à jour, et ce à la fin de son traitement.

URN : /transcodifications/batch

Verbe HTTP : POST

Codes résultat :

- 200 : opération réussie
- 500 : problème de serveur

Diagramme de séquence :

Voir Annexe 4.4.9

4.4.10 Conclusion de l'Architecture détaillée

La section sur l'architecture détaillée du projet RMOD clôture une phase cruciale de conception, établissant une structure claire et fonctionnelle pour le système. Cette architecture, avec ses listes statiques, ses objets métiers et ses services API, a été soigneusement conçue pour répondre aux exigences complexes de la réconciliation des données de véhicules chez CGI Finance. Elle constitue le fondement sur lequel s'appuiera la phase de développement, où nos concepts et structures seront mis en œuvre.

Cette transition de la planification détaillée au développement pratique est une étape essentielle, marquant le passage de la théorie à l'action pour donner vie à la solution RMOD.

4.5 Conclusion sur la conception

Avec l'achèvement de la partie de conception du projet RMOD, nous avons posé des bases solides et détaillées pour la phase de développement qui va suivre. Cette étape de conception a été cruciale pour clarifier et définir les divers éléments techniques et fonctionnels qui composeront notre système. Les listes statiques, les objets métiers, ainsi que les différentes fonctionnalités des services API ont été élaborés pour répondre aux exigences précises du projet.

La conception de l'architecture du système a été une tâche complexe, prenant en compte non seulement les besoins techniques mais aussi les spécificités métier de CGI Finance. L'intégration harmonieuse de cette architecture dans l'environnement existant du SI de CGI Finance est un gage de la réussite future de l'application.

Alors que nous nous préparons à entrer dans la phase de développement, tous ces éléments de conception fourniront un guide détaillé pour la mise en œuvre technique. Cette phase de développement est l'étape suivante où nos plans et nos concepts prendront vie, transformant notre vision en une solution fonctionnelle et efficace pour la réconciliation des données de véhicules. Nous aborderons cette phase avec une compréhension claire de nos objectifs et une stratégie bien définie pour les atteindre.

5 DEVELOPPEMENT

5.1 Introduction au développement

5.1.1 Objectifs

La phase de développement du projet RMOD constitue le cœur de notre démarche, transformant la vision théorique établie dans la phase de conception en une application fonctionnelle et robuste. Cette étape cruciale vise plusieurs objectifs clés :

- **Une réalisation conforme à la conception :** Chaque composant devra être fidèle aux spécifications techniques établies, garantissant que l'application répond aux besoins métier et techniques de CGI Finance.
- **Qualité et résilience du code :** Nous visons à produire un code de haute qualité, caractérisé par sa robustesse et sa maintenabilité. La structuration de celui-ci est donc primordiale, ainsi que l'utilisation de différents design pattern, dans le but d'appliquer au maximum les principes S.O.L.I.D.
- **Développement de tests :** Une attention particulière sera apportée également à l'écriture des tests unitaires et d'intégration, afin de s'assurer que chaque composant fonctionne correctement individuellement, et en conjonction avec les autres parties du système. De plus, CGI Finance exige une couverture de code d'au minimum 93% afin d'avoir l'autorisation de déployer en production.
- **Optimisation des performances :** L'application a pour objectif d'être aussi performante que les outils actuels en temps de réponse, afin de ne pas dégrader l'expérience utilisateurs de ses futurs clients. Ainsi, l'optimisation sera un autre grand défi à relever afin de pouvoir déployer en production.
- **Adaptabilité et scalabilité :** Le développement devra en outre prendre en compte la nécessité d'adapter et d'évoluer l'application au fil du temps. Cela signifie de concevoir des composants modulaires et flexibles qui peuvent être facilement mis à jour ou étendus pour répondre à des besoins futurs ou à des changements dans l'environnement technologique ou métier.

En résumé, la phase de développement vise à concrétiser la vision de la conception en une application performante et fiable, prête à être déployée dans l'environnement de production de CGI Finance.

5.1.2 Méthodologie

La méthodologie employée dans le développement du projet RMOD s'inspire des principes de l'AGILE, adaptée pour répondre aux exigences spécifiques et à l'environnement de travail de CGI Finance. Cette approche a favorisé une flexibilité accrue et une meilleure réactivité face aux exigences changeantes du projet. Cette méthodologie peut être résumée de la façon suivante :

- **Daily Meetings :** Chaque journée de travail commençait par une réunion quotidienne, un « daily meeting ». Ces courts rassemblements matinaux étaient l'occasion de faire le point sur les progrès de chacun, d'identifier les obstacles éventuels et de coordonner les efforts de l'équipe.
- **Reviews réguliers avec le Chef de Service et le Tech Lead :** Toutes les deux semaines en moyenne environ, des sessions de revue étaient organisées avec le chef de service et le tech lead de la DSQUAD. Ces rencontres servaient à examiner les avancées réalisées, à évaluer les résultats obtenus et à ajuster les plans si nécessaire. Cette interaction régulière a permis de maintenir un alignement étroit avec les objectifs du projet et d'assurer une communication fluide et efficace.
- **Utilisation de JIRA pour la communication inter-équipes :** JIRA [13], un outil de gestion de projet et de suivi des problèmes, a été utilisé de manière extensive pour faciliter la communication entre les différentes équipes impliquées dans le projet. Cela a permis une traçabilité et une gestion efficaces des tâches, des anomalies, et des demandes de fonctionnalités, assurant ainsi que tous les membres de l'équipe étaient constamment à jour sur l'état du projet.

Cette combinaison d'éléments agiles avec des processus structurés a fourni un cadre de travail dynamique et adaptatif, essentiel pour gérer efficacement un projet de cette envergure et complexité. L'accent mis sur la communication régulière et la réactivité face aux changements a non seulement aidé à maintenir le projet sur la bonne voie, mais a également contribué à la création d'un produit final qui répond réellement aux attentes des clients finaux et de l'entreprise.

5.2 Présentation de l'environnement de développement

5.2.1 La plateforme .NET 6

Figure 46 - Logo C# & Microsoft .NET

Au sein de CGI Finance, le choix de la plateforme de développement .NET 6 [14], ainsi que son framework ASP.NET Core [15], qui utilisent C# 10 [16] et Visual Studio 2022 [17] comme outils de développement, s'inscrit dans une logique cohérente avec les besoins spécifiques du secteur de la bancassurance. Voici les raisons principales de cette préférence [18] :

- **Sécurité et Conformité :** .NET 6 offre des capacités de sécurité avancées, un aspect crucial pour la gestion des données sensibles dans la bancassurance. La conformité avec les normes réglementaires strictes est facilitée grâce à ces fonctionnalités de sécurité intégrées.
- **Performance et Fiabilité** : Ces outils permettent de développer des applications à la fois performantes et fiables, aptes à gérer les transactions et les volumes de données conséquents, typiques dans le secteur financier.
- **Interopérabilité et Polyvalence** : La capacité de .NET 6 à fonctionner sur diverses plateformes favorise l'interopérabilité dans les environnements informatiques complexes de la bancassurance. C# 10 et Visual Studio renforcent cette polyvalence, permettant un développement agile et une intégration aisée.
- **Écosystème et Communauté** : Le support robuste de Microsoft et la vaste communauté autour de .NET, C# et Visual Studio fournissent un environnement riche en ressources et en expertise, essentiel pour résoudre rapidement les problèmes et rester à jour avec les meilleures pratiques.

Figure 47 - Logo Visual Studio 2022

Ces choix technologiques reflètent donc non seulement les besoins actuels de CGI Finance, mais ils s'alignent également avec les exigences générales de la bancassurance en termes de sécurité, de performance et d'adaptabilité, assurant ainsi une base solide pour le développement et l'évolution continue de leurs systèmes d'information.

5.2.2 PostgreSQL

CGI Finance a choisi PostgreSQL pour sa base de données, malgré la synergie courante entre .NET et SQL Server, sûrement en raison de plusieurs avantages adaptés à leurs besoins spécifiques :

- **Coût-Efficacité** : En tant que système de gestion de base de données open source, PostgreSQL représente une option économiquement avantageuse par rapport à des solutions comme SQL Server. Cette efficacité en termes de coût est particulièrement pertinente pour une entreprise cherchant à optimiser ses ressources tout en maintenant des performances élevées.
- **Performances de Requête Optimisées** : PostgreSQL est réputé pour sa capacité à gérer efficacement les requêtes complexes, un atout essentiel pour le traitement des importantes transactions et volumes de données dans le secteur financier.
- **Flexibilité et Évolutivité** : Sa nature open source confère à PostgreSQL une grande flexibilité, permettant des personnalisations et des extensions adaptées aux besoins évolutifs de CGI Finance.
- **Interopérabilité** : PostgreSQL, bien qu'étant un choix moins conventionnel dans l'écosystème .NET, s'intègre efficacement avec une variété de technologies dont celle-ci.
- **Communauté et Support** : L'ample communauté autour de PostgreSQL assure un environnement riche en ressources et en soutien, crucial pour une réponse rapide aux problèmes et l'adoption des meilleures pratiques.
- **Support du JSONB** : Enfin, PostgreSQL excelle dans la gestion des données JSON grâce à son format JSONB. Cette capacité est cruciale pour le framework interne de CGI Finance, qui utilise intensivement le JSONB pour le stockage et la manipulation des données.

Figure 48 - Logo PostgreSQL

Ce choix technologique s'aligne parfaitement avec les stratégies de gestion de données de CGI Finance, ce qui permet une harmonisation dans leur gestion, en particulier avec le framework interne qui sera le sujet de la prochaine partie.

5.2.3 Le framework interne CGI Finance

Dans un objectif de standardisation des développements, CGI Finance a créé son propre framework interne, surcouche de ASP.NET Core faisant partie de la plateforme .NET.

Basé sur le design pattern « Mediator » [19], il permet de réduire les communications complexes ou les dépendances directes entre les objets et les classes, en les centralisant en un seul point.

Il intègre également un système de gestion de base de données, avec pour chaque objet métier enregistrés, la création de trois tables (une table de stockage, une table historique et une table audit) permettant l'historisation des modifications. Ces objets sont enregistrés en JSONB dans une colonne, et ainsi on a une standardisation du stockage quelques soient les applications ou les structures des objets à enregistrer.

Il intègre également un système de logs standardisé enregistrant automatiquement chaque requête, et qui les envoie, après un minimum de configuration, directement sur les environnements ELK en place.

Bien que le framework offre des avantages significatifs en termes de standardisation et d'efficacité, il nécessite une courbe d'apprentissage, exacerbée par un manque de documentation. Cette situation a souvent requis de consulter directement le chef de projet responsable de ce framework pour des éclaircissements.

5.2.4 Azure DevOps

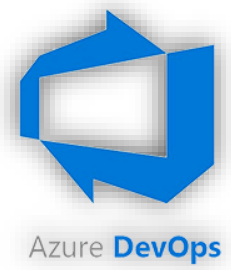


Figure 49 - Logo Azure DevOps

Azure DevOps est une suite d'outils proposée par Microsoft qui facilite la gestion du cycle de vie complet du développement logiciel. Dans le cadre de RMOD, les aspects suivants ont été utilisés :

- **Gestion de code source avec Azure Repos :** Cette fonctionnalité héberge les dépôts Git ce qui permet le versioning et la collaboration sur le projet.
- **Intégration et Livraison Continue (CI/CD) avec Azure Pipelines :** Cette fonctionnalité a permis d'automatiser les processus de build, de test et de déploiement de l'application.

En résumé, l'utilisation d'Azure DevOps dans le projet RMOD s'est concentrée sur ces deux aspects (qui sont loin d'être les seuls proposés par cette suite d'outils). Et dans le cas du CI/CD l'automatisation du contrôle de la qualité du code fut également implémenté, c'est le sujet de la prochaine section sur SonarQube.

5.2.5 SonarQube

Figure 50 - Logo SonarQube

SonarQube [20] est un outil d'analyse de code source qui joue un rôle crucial dans le maintien de la qualité et de la sécurité du code dans les projets de développement logiciel. Au sein du projet RMOD chez CGI Finance, SonarQube a été intégré pour plusieurs raisons clés :

- **Analyse de la Qualité du Code :** SonarQube examine le code source pour détecter les bugs, les vulnérabilités et les "code smells" (mauvaises pratiques de codage). Cela a permis de s'assurer que le code développé pour RMOD était non seulement fonctionnel mais aussi propre et bien structuré.
- **Intégration avec Azure DevOps :** SonarQube s'intègre parfaitement avec Azure DevOps, utilisé dans le cadre de notre projet. Cette intégration a permis une analyse et un feedback automatique sur la qualité du code à chaque push et chaque build depuis le dépôt Git.
- **Amélioration Continue :** SonarQube contribue à l'amélioration continue du code en fournissant des rapports détaillés et des recommandations. Ces informations ont été utilisées pour optimiser en permanence le code de RMOD, réduisant ainsi les risques de bugs et améliorant les performances globales.

Pour conclure, l'exigence de CGI Finance pour tout déploiement en production d'une application est d'atteindre une note de « A » dans chaque domaine d'analyse de SonarQube. C'était donc d'emblée notre objectif.

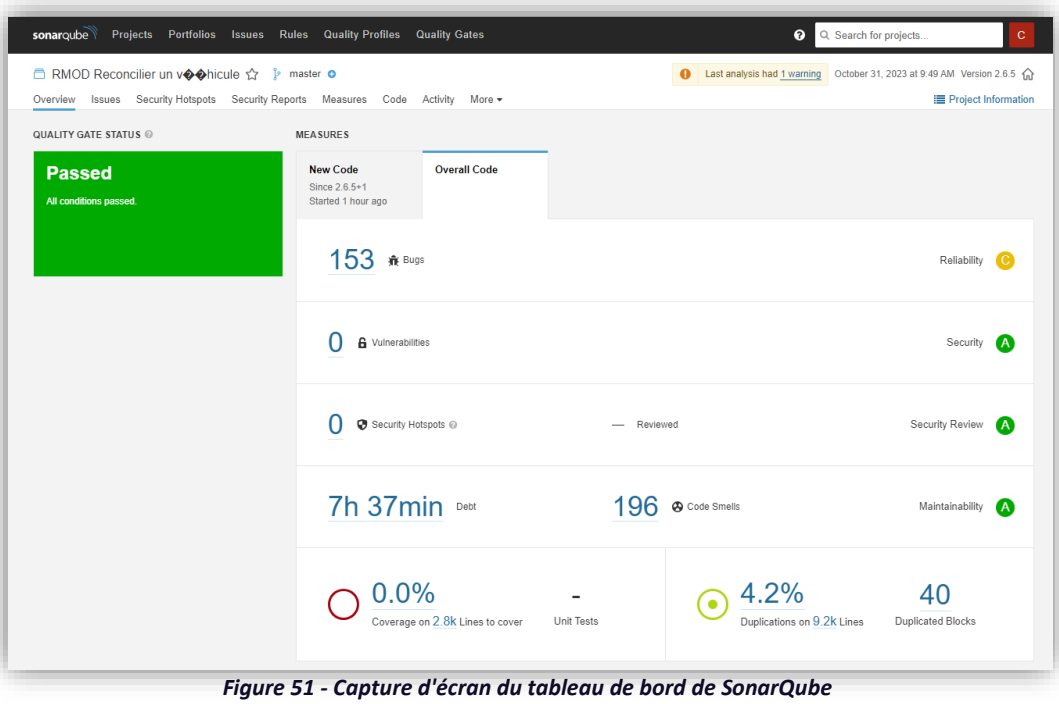


Figure 51 - Capture d'écran du tableau de bord de SonarQube

5.2.6 Liquibase

Figure 52 - Logo Liquibase

Liquibase [21] est un outil de gestion de changements de base de données qui a joué un rôle crucial dans le processus de développement du projet RMOD chez CGI Finance. Son utilisation s'inscrit dans une stratégie de contrôle rigoureux et de gestion efficace des évolutions de la base de données. Voici les aspects clés de son utilisation :

- **Autonomie en Environnement de Développement** : Nous avions tout pouvoir sur la base de données de développement (DEV). Ainsi, dans cet environnement, l'utilisation d'un outil tel que Liquibase n'était pas nécessaire. Car nous pouvions faire n'importe quelle opération directement sur la base de données sans avoir nullement de compte à rendre. Ceci dans l'esprit d'offrir l'autonomie nécessaire à l'innovation.
- **Contrôle Restreint en Environnement de Recette** : Dans l'environnement de recette (REC), nous ne disposions pas « normalement » des droits d'accès direct, ce qui fut le cas longtemps. Mais par souci de praticité notamment pour des interventions rapides pendant la phase de testing, nous avons pu les avoir. Néanmoins, pour une évolution durable, Liquibase était privilégié afin d'assurer la cohérence et la traçabilité entre les environnements.
- **Gestion Rigoureuse des Évolutions en Production** : Sur la base de données de production (PROD), nous n'avions que des droits de lecture. Toutes les modifications de la base de données doivent passer par Liquibase. Ce processus garantit que les changements sont appliqués de manière standardisée et réversible, minimisant ainsi les risques d'erreurs et assurant la traçabilité des évolutions.
- **Traçabilité et Réversibilité** : Un atout majeur de Liquibase est sa capacité à suivre précisément chaque modification apportée à la base de données, permettant ainsi un retour en arrière si nécessaire. Cette fonctionnalité est essentielle pour maintenir la stabilité et la fiabilité de la base de données en production.
- **Facilitation des Déploiements** : Liquibase facilite le processus de déploiement en automatisant l'application des scripts de modification de la base de données, le tout hébergé sur Azure DevOps.

En conclusion, l'utilisation de Liquibase, obligatoire chez CGI Finance pour des raisons de sécurité, de traçabilité et de normalisation,

5.3 Développement des composants clés

5.3.1 Architecture de la solution

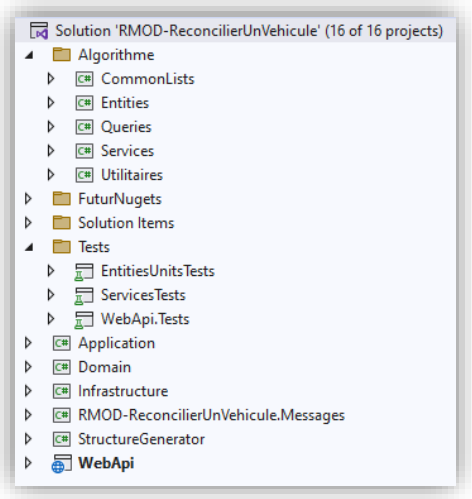


Figure 53 - Organisation des projets de la solution

La solution est architecturée suivant trois grandes parties :

- Celle regroupant tous les projets nécessaires au framework interne CGI
- Celle regroupant tous les projets de tests (dossier Tests)
- Celle regroupant les services qui contiendront la logique, qui a été conçue pour être totalement indépendante (dossier Algorithme)

Ainsi, avec cette approche, nous pouvons si besoin changer le mode d'utilisation des services de réconciliation et mise en qualité sans aucune modification du code (avec par exemple une autre surcouche API que le framework interne CGI).

Le point d'entrée de l'API est donc le projet « WebApi », et les différents points d'entrée pour la logique seront regroupés dans le projet Algorithme/Services, ainsi que le montre le schéma de dépendance simplifié des projets de la solution ci-dessous.

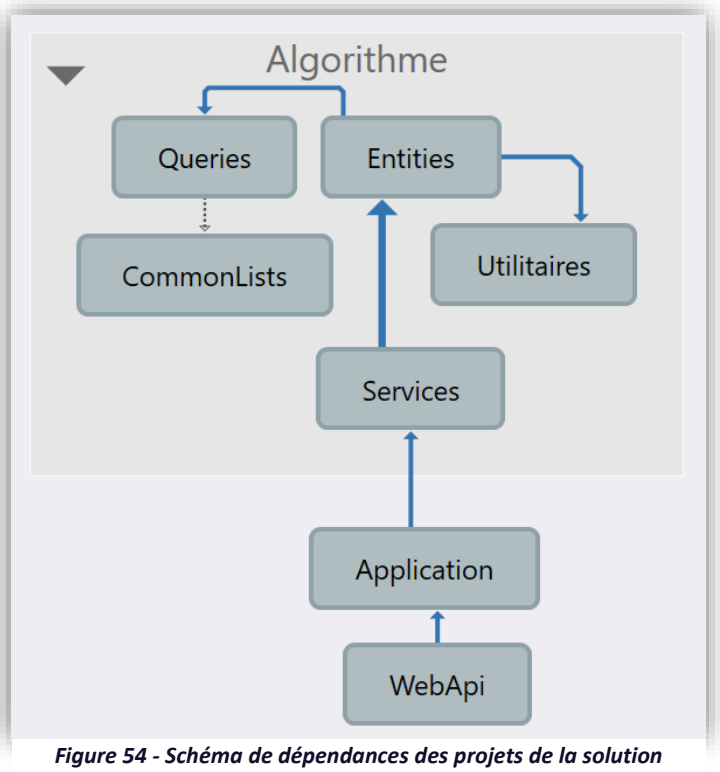


Figure 54 - Schéma de dépendances des projets de la solution

5.3.2 Définition des endpoints

Les différents endpoints, correspondant chacun à une fonctionnalité de la phase de conception, sont définis suivant la syntaxe imposée par le framework CGI.

```
public class ReconcilierUnVehiculeDescriptor : WebApiTypeDescriptors
{
    private const string accederAuxReconciliationsModeleVehicule_root = "reconciliations";
    private const string reconcilierModeleVehicule_root = "reconcilierModeleVehicule";
    private const string reconcilierModeleImmatriculation_root = "reconcilierModeleImmatriculation";
    private const string retranscrireModeleVehicule_root = "retranscrireInformationsVehicule";

    private const string accederStrategiesReconciliationModeleVehicule_root =
        "strategiesReconciliationModeleVehicule";

    private const string accederTranscodificationsInformationsVehicule_root = "transcodifications";

    public ReconcilierUnVehiculeDescriptor()
    {
        AjoutReconciliationModeleVehiculeDescriptors();
        AjoutAccederReconciliationModeleVehiculeDescriptors();
        AjoutAccederStrategiesReconciliationModeleVehiculeDescriptors();
        AjoutAccederTranscodificationsInformationsVehiculeDescriptors();
        AjoutTranscoBatchDescriptor();
    }

    private void AjoutTranscoBatchDescriptor()
    {
        AddDescriptor<TranscoBatchCommand>(c => c
            .ForRoute($"/{accederTranscodificationsInformationsVehicule_root}/batch")
            .ForVerb(HttpMethod.Post)
            .WithSuccessHttpCode(HttpStatusCode.OK)
        );
    }

    private void AjoutAccederTranscodificationsInformationsVehiculeDescriptors()
    {
        AddDescriptor<ConsulterTranscodificationInformationVehiculeQuery>(c => c
            .ForRoute($"/{accederTranscodificationsInformationsVehicule_root}/{{Identifiant}}")
            .ForVerb(HttpMethod.Get)
            .ForLocation(WebApiTypeDescriptor.ParameterLocation.QUERY)
            .ProduceResponse(HttpStatusCode.NoContent)
            .ProduceResponse(HttpStatusCode.BadRequest)
            .ProduceResponse(HttpStatusCode.InternalServerError)
            .WithSuccessHttpCode(HttpStatusCode.OK)
        );
```

Figure 56 - Définition des endpoints de l'API

Ainsi, dans un seul et même endroit, tous les endpoints de l'API sont configuré, avec leur verbe http, la localisation des paramètres (body ou query), leur route ainsi que les différents types de retour d'erreur. Ceci montre donc bien comment est utilisé le design pattern Mediator dans la construction du framework.

De plus, cette approche permet, pour une meilleur lisibilité, l'organisation des endpoints par service comme dans la phase de conception.

```
        services.AddSwaggerGen(setup =>
        {
            setup.SwaggerDoc(_serviceName, new OpenApiInfo { Title = _serviceName, Version = "v1" });
            var messageAssembly = typeof(ReconcilierUnVehiculeDescriptor).Assembly;
            //setup.IncludeXmlComments(messageAssembly.Location.Replace(".dll", ".xml"));
            setup.SchemaFilter<PropertyAlreadySettedSchemaFilter>();
        });

        services.ConfigureSwaggerGen(c =>
            c.SwaggerDoc("v1", new OpenApiInfo { Title = "Service exposant les opérations de recherche
complexes à partir d'informations de véhicule", Version = "1.0" })
        );
```

Figure 55 - Ajout de Swagger

Enfin, dans un objectif de documentation, Swagger a été implémenté, et sur chaque objet d'entrée pour chaque endpoint, une description de celui-ci a été mis en place, organisé par catégorie, ce qui permet également de regrouper dans Swagger les endpoints suivant leurs services (voir Annexe 5.3.2).

```
    [Category("Service Réconcilier un véhicule")]
    [Description(
        "Permet de déterminer le modèle du véhicule référencé dans le référentiel catalogue
constructeur à partir d'informations brutes sur le véhicule")]

    public class ReconcilierModeleVehiculeCommand :
ICommand<CreerCommandResult<ReconciliationModeleVehiculeMessage>>
    {
        public string CodeStrategieReconciliation { get; set; }
        public InfosVehicule InfosVehicule { get; set; }
    }
```

Figure 57 - Exemple de définition d'un objet "Demande" avec sa description

5.3.3 Dépendance de type des différentes opérations

Dans le diagramme de dépendance de types présent en Annexe 5.3.3, nous pouvons voir comment interagissent les types gérant les différents endpoints de l'API.

En effet, afin de diminuer la redondance de code, et vu que des actions sont dépendantes d'autres, certains endpoints vont appeler directement les types nécessaires à leur fonctionnement.

Sur le diagramme, les flèches bleues pleines correspondent aux dépendances directes, et les flèches rouges pointillées les injections de dépendances.

De plus, par convention, les endpoints utilisant un verbe http « GET » seront nommés en « QueryHandler », et les « POST » en « CommandHandler ».

On peut donc ici se rendre compte que, par exemple, la réconciliation est dépendante du service externe IAccesANTSRequests (qui gérera l'appel à l'API de récupération des informations par plaque), du ISIVArgusService (qui appellera le service externe SIV Argus en cas d'échec), mais également des autres services internes comme la réconciliation par caractéristiques, de la retranscription (mise en qualité) et de la consultation des transcodifications.

Le service de réconciliation par immatriculation utilise donc un parcours complet, de la récupération des données via l'immatriculation, à la réconciliation en passant par la mise en qualité des données (la récupération des transcodifications et la retranscription).

A noter que ce diagramme se cantonne à la surcouche du framework CGI et ne prend pas en compte tous les types de la solution.

5.3.4 Développement des fonctionnalités de création

Concernant le développement de la logique des fonctionnalités de création, elles sont toutes similaires dans leur fonctionnement. Et ne feront intervenir que du code géré par le framework CGI.

Nous allons donc prendre pour exemple la fonctionnalité de création des transcodifications pour illustrer l'ensemble.

Dans la plupart des cas dont celui-ci, la logique imposée par le framework CGI se décompose en quatre parties :

- L'objet « Command » qui correspond à un objet « Demande » vu dans la partie conception. (Annexe 5.3.4.a)
- L'objet « CommandResult » qui n'est autre que l'objet « Résultat » vu dans la partie conception. (Annexe 5.3.4.b)

- Le « CommandHandler » qui prendra en un objet « Command » en paramètre, renverra un « CommandResult » en résultat et effectuera la logique tel un contrôleur traditionnel. (Annexe 5.3.4.c)
- La classe « Aggregate » qui gérera la partie persistance en base de données. (Annexe 5.3.4.d)

On peut de plus remarquer dans l'objet « Aggregate », qui a pour but d'enregistrer en base de données un objet métier (appelé ici « Message » par convention), l'apparition d'un objet « Audit » et un objet « CycleDeVie ». Ce sont ces deux objets qui serviront au framework pour effectuer l'historisation des objets métiers.

5.3.5 Développement des opérations de passage en validée

La structure générale de la logique de ces opérations étant très similaires aux opérations de création, nous ne nous attarderons pas dessus outre mesure.

Néanmoins, le cas de la validation des transcodifications est intéressant car, en se référant à la conception, il intègre une étape supplémentaire : la détermination modèle/sous-modèle.

En effet, il est vérifié que, si la valeur passée en paramètre (dans l'objet « Command ») correspond à un sous-modèle, dans ce cas, le type de la transcodification sera sous-modèle. Dans le but d'avoir une information plus précise afin de rendre plus précise la réconciliation ultérieure.

Nous récupérons donc la liste des sous-modèles de la marque dans le catalogue constructeur, et nous vérifions si la valeur envoyée pour validation correspond à l'une d'entre elle.

Voir les annexes :

- Récupération de la liste des sous-modèles par marque (Annexe 5.3.5.a).
- La classe Passer transcodification en validée Command Handler (Annexe 5.3.5.b).
- Une méthode de la classe Transcodification Aggregate (Annexe 5.3.5.c).

5.3.6 Développement des opérations de consultation

Les opérations de consultations sont également très similaires, à la différence que, comme un verbe http « GET » est utilisé, l'objet utilisé n'est pas un « CommandHandler » mais un « QueryHandler » qui prend en paramètre une « Query » et en résultat un « QueryResult ».

De plus, il est nécessaire de par le fonctionnement du framework d'utiliser une requête SQL classique, sans passer par un ORM tel qu'Entity Framework.

Toujours en prenant pour exemple les transcodifications, nous avons donc :

- La classe Consulter Transcodifications QueryHandler (Annexe 5.3.6.a).
- Une méthode de la classe Transcodification Aggregate (Annexe 5.3.6.b).

5.3.7 Développement des opérations de passage en abandonnée

En ce qui concerne les opérations de passage en abandonnée, elles sont extrêmement similaires aux précédentes, ainsi, nous n'entrerons pas dans les détails.

Il y a une vérification si la transcodification existe et qu'elle n'est pas déjà abandonnée, et si tel est le cas, il n'y a que le changement de son statut à effectuer. (Annexe 5.3.7).

C'est exactement le même principe dans le cas des stratégies de réconciliations.

5.3.8 Transformation des caractéristiques en dimensions : la fonction Reduce()

Nous allons maintenant aborder un élément central du fonctionnement de l'algorithme tel que nous l'avons conçu : la fonction Reduce(). En effet, celle-ci sera utilisée pour la réconciliation, mais également pour la retranscription (en particulier l'enrichissement).

La fonction est portée par la classe « InfosVehicule », qui va contenir les propriétés représentant les caractéristiques de véhicule en entrée de l'algorithme. Ce choix a été fait afin d'utiliser la réflexivité dans cette classe pour, en conjonction avec l'utilisation du design pattern Factory [22], créer de façon dynamique les différentes dimensions, et la quasi-intégralité du code réutilisable dans l'éventualité d'une évolution future qui ajouterait une nouvelle dimension à prendre en compte (voir Annexe 5.3.8.a).

De plus, c'est au sein de cette fonction qu'est également effectuée la vérification de la présence, dans le cas où c'est nécessaire, des caractéristiques obligatoires marque, modèle et date de mise en circulation.

En ce qui concerne la DimensionFactory, elle a pour rôle de créer chaque dimension, de lui attribuer son type et sa valeur. De, plus, en cas de besoin, elle y attribuera la pondération issue de la stratégie associée. (voir Annexe 5.3.8.b).

A noter que la classe Dimension porte en elle la fonction qui calcule les distances, et qui fait intervenir une autre Factory, sur laquelle nous reviendrons plus tard. (voir Annexe 5.3.8.c).

Et enfin, la fonction Reduce() récupère les dimensions ainsi créées et les renvoie dans une liste de dimensions qui correspond à un véhicule.

5.3.9 Développement de la retranscription

En ce qui concerne les opérations de retranscription, qui regroupent la transcodification automatique et l'enrichissement, il y eut un débat au niveau de la conception, avec l'architecte S.I. d'un côté et nous de l'autre.

En effet, nous étions en désaccord avec l'architecte sur le regroupement de ces deux opérations au sein du même endpoint. Car les traitements au niveau technique sont totalement différents, augmentant ainsi la complexité du code, et par conséquent sa simplicité et maintenabilité.

Le point de vue, théorique, de l'architecte se tenait car ces deux opérations font partie de la mise en qualité des données automatisée, et que, pour lui, la division en deux endpoints différents rajouterait de la complexité au niveau des clients qui consommeront l'API.

De notre point de vue, la séparation était nécessaire, d'un point de vue technique, pour garder l'unicité des responsabilités (1 endpoint = 1 opération). Et nous avions la conviction que d'un point de vue client, la réunion des deux opérations dans ce même endpoint allait provoquer de la confusion en n'étant pas très clair sa finalité…

Après quelques échanges, l'architecte a finalement imposé son point de vue qui s'intégrait de façon plus cohérente dans le S.I. pour lui, et nous avons donc été contraint de suivre cette solution.

Nous n'allons cependant pas rentrer dans les détails au niveau du endpoint de la façon dont a été géré cette difficulté, car elle ne nous convient pas, considérant qu'elle n'est pas très « propre » et provoque un code non optimisé. Nous allons donc passer directement à la façon dont sont développées les deux fonctionnalités, qui, dans la partie algorithme, a bien des services séparés. Services qui sont les points d'entrée indépendants des différentes opérations d'un point de vue technique.

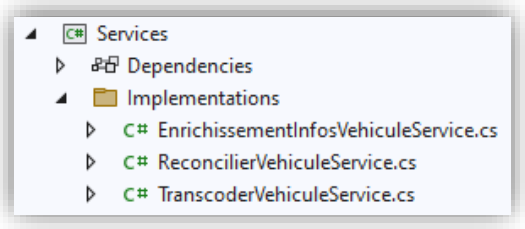


Figure 58 – Hiérarchie des services dans la solution

5.3.10 Développement de la transcodification automatique

Le service de transcodification automatique est très simple n'ayant que trois méthodes à utiliser suivant le type de transcodification à effectuer. (voir Annexe 5.3.10.a).

Un middleware a été mis en place (TranscoProcess), avec un système de cache, qui va effectuer les opérations spécifiques de chaque méthode comme récupérer les données nécessaires dans le catalogue constructeur qui serviront de base de comparaison. Et ensuite va tout envoyer vers l'objet qui effectuera le processus de transcodification automatique de façon générique, quelque soit le type de transcodification à trouver (SuggestTranscos). Le cache permettra de ne pas refaire la requête en base de données si des données récentes sont déjà présentes, améliorant ainsi les performances. (voir Annexe 5.3.10.b).

Et enfin, la classe SuggestTranscos, qui va effectuer les recherches des valeurs transcodées et les porter dans ses propriétés. C'est d'ailleurs l'objet de retour du service.

Il comporte deux propriétés pour chaque test effectué, un booléen pour déterminer si le résultat trouvé est considéré comme acceptable, et la valeur correspondante. C'est ici que les règles métiers de l'égalité et du « contenu » vont s'appliquer pour une transcodification automatique considérée comme sûre, et c'est ici également que seront trouvées les suggestions qui devront être validées par un expert métier, si tant est qu'il y ait des correspondances. (voir Annexe 5.3.10.c).

5.3.11 Développement de l'enrichissement des données

Le service effectuant l'enrichissement des données à partir du champ version est beaucoup plus complexe. Il a pour tâche de trouver des correspondances dans ce champ et de créer les dimensions avec les données correspondantes.

Il a été conçu également pour trouver le modèle dans le cas où nous n'aurions pas la donnée initialement.

La classe d'entrée du service reste relativement simple, avec une seule méthode qui va gérer le processus au plus haut niveau. Elle va de plus utiliser la fonction ReduceVehicule() (qui appelle la fonction Reduce() vue plus haut) pour transformer les caractéristiques de véhicules en dimensions. Et va également vérifier si le champ version est nul ou vide. (voir Annexe 5.3.11.a).

En utilisant à nouveau de design pattern Factory, l'algorithme va déterminer quelles données sont potentiellement trouvables parmi le modèle, la puissance DIN, la finition, le nombre de places, le nombre de portes ou le sous-modèle.

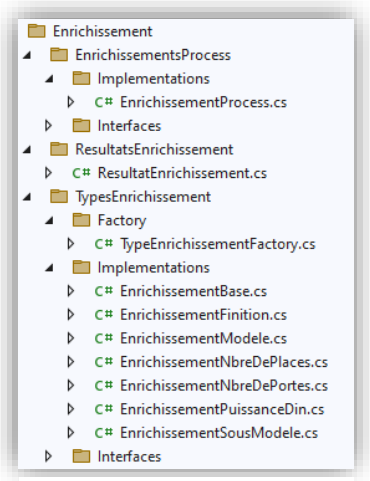


Figure 59 - Hiérarchie du service d'enrichissement

A noter l'utilisation de l'héritage ici, avec la classe EnrichissementBase qui va contenir des traitements génériques qui pourront être utilisées par les classes filles. (voir Annexe 5.3.11.b).

Chaque type de données à extraire a donc ici son propre processus spécifique, avec utilisation dans le cas des opérations similaires, des méthodes définies dans la classe mère.

En Annexe 5.3.11.c, nous pourrons voir un exemple simple avec l'enrichissement du nombre de places. Par souci de concision, ce sera le seul exemple spécifique que nous donnerons sur les opérations d'enrichissement, car chaque type peut avoir différentes opérations plus ou moins complexes, ce qui prendrait énormément de place et de temps à décrire de façon exhaustive ici.

5.3.12 Développement de la réconciliation par caractéristiques

Après avoir vu les opérations de mise en qualité de données, voici maintenant le cœur du système avec la réconciliation par caractéristiques.

La classe service (voir Annexe 5.3.12.a) en tant que telle sera dans ce cas celle qui orchestrera toutes les opérations afin d'obtenir une réconciliation qui sont dans l'ordre :

- Transformation des caractéristiques du véhicule à tester en liste de dimensions
- Récupération des véhicules candidats dans le catalogue constructeur via marque, date mise en circulation et modèle (ou sous modèle) (Annexe 5.3.12.b)
- Transformation des caractéristiques véhicule des candidats en liste de dimensions
- Calcul des distances (voir partie suivante)
- Suppression des distance si elles sont trop élevées, utilisée pour la finition pour éviter les biais quand on a une valeur de distance très élevée > 0.6 (et donc une valeur considérée comme fausse ou incohérente) (Annexe 5.3.12.c)
- Tri des candidats ayant la même énergie que le véhicule à tester, rendant obligatoire la valeur de l'énergie dans le véhicule à tester (Annexe 5.3.12.d)
- Sélection du prix catalogue dont la période de commercialisation comprend, ou est au plus proche de la date de mise en circulation, afin d'avoir une valeur unique malgré la variabilité dans le catalogue constructeur (Annexe 5.3.12.e)

- Suppression des doublons en sélectionnant la ligne qui a la plus faible distance générale, afin de corriger le souci de la variabilité de certaines informations dans le catalogue constructeur (Annexe 5.3.12.f)
- Application du seuil de distance maximal défini dans la stratégie (Annexe 5.3.12.g)
- Tri par distance pour établir la hiérarchie des résultats (Annexe 5.3.12.h)
- Application du paramètre de la valeur minimale de reprise défini dans la stratégie, qui sélectionnera le véhicule avec le prix le plus bas s'il est activé (Annexe 5.3.12.i)
- Construction et retour de l'objet de résultat

5.3.13 Calcul des distances

Nous avons mis de côté le calcul des distances pour cette partie distincte, car étant réellement le moteur de l'ensemble de l'algorithme.

Nous avons déjà vu dans une partie précédente, celle sur la fonction Reduce(), que la fonction de calcul de distance était portée par la classe Dimension elle-même.

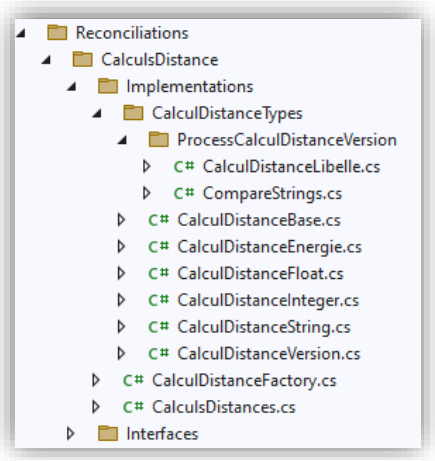


Figure 60 - Hiérarchie des classes servant au calcul des distances

La fonction CalculDistances() appelée dans le service, quant à elle, prend en paramètres la liste des distances du véhicule à tester, ainsi que l'ensemble des listes de distances des véhicules candidats. Elle instancie un nouvel objet qui va gérer la correspondance des dimensions de même type, l'appel de la fonction de calcul de la distance pour la dimension (celle que nous avons déjà rencontré plus haut), ainsi que celle qui mettra à jour la distance générale du véhicule candidat concerné (Annexe 5.3.13.a).

```
public List<IListDimensionsVehicule> CalculDistances(IListDimensionsVehicule dimensionsVehATester,
    List<IListDimensionsVehicule> listVehiculesCandidats)
{
    var calculDistances = new CalculsDistances(dimensionsVehATester, listVehiculesCandidats);
    return calculDistances.MatchingDimensionsAndCalculDistance();
}
```

Figure 61 - La fonction CalculDistances()

La fonction va donc parcourir tous les véhicules candidats, y parcourir toutes ses dimensions, et pour chacune d'entre elle parcourir les dimensions du véhicule à tester. Si elle trouve une correspondance de type de dimension, elle va appeler dans ce cas la fonction CalculerDistance() portée par la dimension du véhicule à tester, qui va utiliser une Factory afin de déterminer le calcul qu'elle devra effectuer (Annexe 5.3.13.b) parmi plusieurs opérations prédéfinies suivant le type de la donnée portée par la dimension.

Nous avons donc six classes de calcul de distances différentes :

- La classe mère qui calculera la distance discrète pour toutes les classes filles qui le souhaiteront. (Annexe 5.3.13.c).
- La classe pour les entiers qui calculera la distance euclidienne normalisée.
- La classe pour les flottants qui calculera également la distance euclidienne normalisée. (Annexe 5.3.13.d).
- La classe pour les chaines de caractères qui calculera la distance de Levenshtein normalisée. (Annexe 5.3.13.e & Annexe 5.3.13.f).
- La classe pour l'énergie qui a besoin d'un traitement spécifique concernant les véhicules hybrides, malheureusement codé en dur, et ensuite une distance de Levenshtein normalisée. (Annexe 5.3.13.g).
- La classe pour le champ version qui calculera la distance de Jaccard normalisée. (Annexes 5.3.13.h à j).

Et enfin, nous avons la fonction CalculDistanceGenerale() qui va incrémenter la distance du véhicule actuelle avec la nouvelle calculée, en y appliquant la pondération correspondante.

```
public class ListDimensionsVehicule : IListDimensionsVehicule
{
    public ListDimensionsVehicule()
    {
        Distance = 0;
    }

    public List<IDimension> Dimensions { get; set; }
    public float Distance { get; set; }
    public IInfosVehicule InfosVehicule { get; set; }

    public void CalculDistanceGenerale(IDimension dimensionReferentiel)
    {
        if (dimensionReferentiel.DistanceDimension != null)
        {
            Distance += dimensionReferentiel.Ponderation *
(float)dimensionReferentiel.DistanceDimension;
        }
    }
}
```

Figure 62 - Classe Liste dimensions d'un véhicule avec la fonction de calcul de la distance générale

5.3.14 Développement de la réconciliation par immatriculation

Le développement de la réconciliation par immatriculation, une fois arrivé à ce stade, va essentiellement consister en un « CommandHandler » qui va utiliser les différents services vu plus haut, avec, de plus, l'utilisation et le développement de services qui vont appeler l'API de récupération des données véhicules via une immatriculation (Annexe 5.3.14.a), et appeler le SIV Argus dans le cas d'échec de la réconciliation (Annexe 5.3.14.b).

En premier lieu, l'appel au service de récupération des informations véhicule par immatriculation est effectué, et la réponse analysée.

```
var resultRequestANTS = await _accesAntsProvider.LaunchRequest();

if (resultRequestANTS.Item1 == null)
{
    if (resultRequestANTS.Item2.Contains("BadRequest"))
    {
        return ObjetMetierErrorHelper.Validation.Add(ErrorConstants.ERR_BADFORMAT,
                "La plaque n'a pas un format valide", "Immatriculation",
                 request.Immatriculation)
            .BuildRequestResult<CreerCommandResult<ReconcilierParImmatriculationCommandResult>>
             ();
    }

    warnings.Add(CheckResultRequestAnts(resultRequestANTS));
    return await AppelSivArgusAsync(warnings, request.Immatriculation);
}

_ = new Vehicule();
Vehicule vehiculeAReconcilier;
if (resultRequestANTS.Item1.Vehicules.Count() > 1)
{
    vehiculeAReconcilier = ChoiceVehicleWithDate(resultRequestANTS.Item1.Vehicules, request);
}
else
{
    vehiculeAReconcilier = resultRequestANTS.Item1.Vehicules.ToList()[0];
}
```

Figure 63 - Appel du service de récupération des informations véhicules par immatriculation

La fonction ChoiceVehicleWithDate() sert à choisir, dans le cas de plusieurs résultats (ce qui est possible comme vu dans l'état de l'art), le véhicule avec la plus ancienne date de carte grise dans le cas d'une plaque définitive, et avec la date de première immatriculation la plus récente dans le cas d'une plaque provisoire.

Ensuite, nous vérifions les transcodifications marque, modèle et énergie. Si les transcodifications marque ou modèle n'existent pas, nous appelons le SIV Argus avec un message d'erreur (dans la fonction CheckTranscoAsync()).

```
var transcoService =
    new ConsulterTranscodificationInformationVehiculeQueryHandler(
        _transcodificationInformationVehiculeDataLayer);
var transcoMarque = await transcoService.Handle(new
        ConsulterTranscodificationInformationVehiculeQuery
{ Identifiant = "MARQUE," + vehiculeAReconcilier?.Marque });

if (transcoMarque == null || transcoMarque.Data == null ||
    transcoMarque.Data.Data.Validation == null ||
    transcoMarque.Data.Data.Validation.ValeurValidee == "")
{
    return await CheckTranscoAsync(warnings, transcoMarque, request, "marque",
                                   vehiculeAReconcilier?.Marque);
}
```

Figure 64 - Appel aux transcodification marque dans la réconciliation par immatriculation

Et enfin, nous lançons la réconciliation par caractéristiques avec les données récupérées et mises en qualité. Si jamais il n'y a pas de résultat, nous appelons le SIV Argus, sinon nous renvoyons les véhicules trouvés après mise en forme suivant les spécifications vues dans la partie conception.

De plus, une stratégie de réconciliation a été définie spécialement pour ce cas, qui porte le code « ANTS ». Elle a été définie après analyse des résultats.

```
var reconciliationResult = await reconcilierService.Handle(new ReconcilierModeleVehiculeCommand
{
    CodeStrategieReconciliation = "ANTS",
    InfosVehicule = new RMOD_ReconcilierUnVehicule.Messages.GroupeInformations.InfosVehicule
    {
        DatePremiereMec = vehiculeAReconcilier.Datepremimmat,
        Marque = transcoMarque.Data.Data.Validation.ValeurValidee,
        Modele = transcoModele.Data.Data.Validation.ValeurValidee,
        Energie = energie,
        Carrosserie = vehiculeAReconcilier.Carrosserie,
        EmissionCo2 = (float?)vehiculeAReconcilier.Co2,
        NombreDePlaces = vehiculeAReconcilier.Nbplacesassises,
        PuissanceFiscale = vehiculeAReconcilier.Puissanceadmin,
        Poids = (float?)vehiculeAReconcilier.Poidsavide,
        SousModele = sousModele
    }
});

if (reconciliationResult == null || !reconciliationResult.Data.Data.Candidats.Any())
{
    return await AppelSivArgusAsync(warnings, request.Immatriculation);
}

var resultReconciliation = await FormatResponseWithReconciliationAsync
                (warnings,resultRequestANTS, vehiculeAReconcilier, transcoModele,
                 transcoEnergie, reconciliationResult);

return RequestResult.Success(new CreerCommandResult<ReconcilierParImmatriculationCommandResult>
                    { Data = resultReconciliation, Warnings = warnings });
```

Figure 65 - Lancement de la réconciliation par caractéristiques dans le CommandHandler

5.3.15 Critique

Le code est particulièrement verbeux par endroit, et ceci étant dû à la structure des objets imposés par le framework CGI.

Néanmoins, il aurait très largement pu gagner en lisibilité si, au lieu d'appeler les CommandHandler pour les appels aux transcodifications (comme montré à la figure 65) et à la réconciliation par caractéristiques, nous avions appelé directement les services de l'algorithme. Ce qui était techniquement faisable, et d'ailleurs la séparation des responsabilités entre le framework et l'algorithme avait été de base pensé pour cela.

Ce cas arrive, de plus, dans chaque CommandHandler, ou QueryHandler, qui appelle d'autres parties de l'API.

Ainsi, nous reconnaissons ici une erreur de jugement quant à l'écriture du code, ce qui nuit à sa maintenabilité et sa compréhension par des tiers, en plus d'y ajouter une couche d'abstraction inutile qui pourrait nuire aux performances…

5.4 Tests et validation

5.4.1 Les tests unitaires

Afin de garantir dans le temps la maintenabilité et la cohérence des résultats, de nombreux tests unitaires ont été écrit afin de sécuriser les évolutions futures.

Ces tests sont surtout ici appliqués aux classes d'enrichissement, les différentes factories et aux calculs de distances.

Ils vont donc utiliser des données réelles, et même des cas limites, pour tester la robustesse de chaque composant au cœur de l'algorithme du projet.

Pour leur réalisation, le framework de test NUnit a été utilisé, qui est l'un des deux principaux existant en .NET.

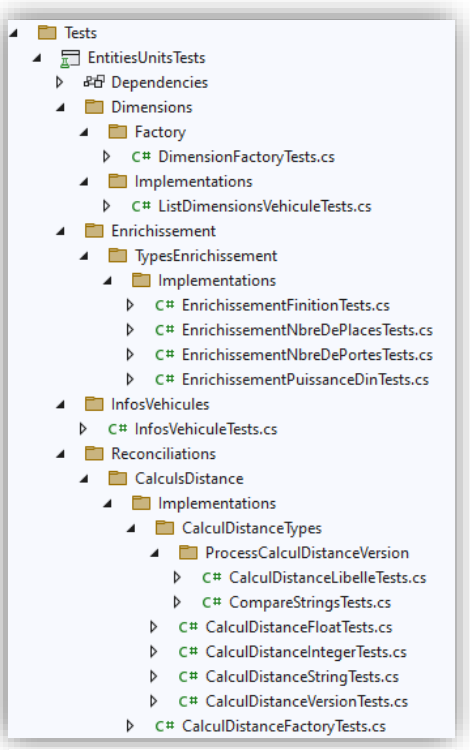


Figure 66 - Hiérarchie des tests unitaires du projet

```
[TestCase("BlueHDi 120ch Live S&S EAT6", 120)]
[TestCase("500h 359ch Executive 4WD", 359)]
[TestCase("180 d Inspiration", null)]
[TestCase("180 136ch Progressive Line", 136)]
[TestCase("A 180 d 116ch Progressive Line 7G-DCT", 116)]
[TestCase("200 d Business Edition 7G-DCT", null)]
[TestCase("220 d 194ch Fascination 9G-Tronic Euro6d-T", 194)]
[TestCase("180d 116ch Progressive Line 7G-DCT", 116)]
[TestCase("180 d 116ch AMG Line 7G-DCT", 116)]
[TestCase("0.9 TCe 95ch Zen - 20", 95)]
[TestCase("1.2 PureTech 110ch Allure S&S EAT6", 110)]
[TestCase("1.5 dCi 110ch energy Limited Euro6 2015", 110)]
[TestCase("1.6 TDI 115ch FAP Trendline Euro6d-T 5p", 115)]
public async Task Testing_With_Reals_Libelle_Version(string actual, int? expected)
{
    // Arrange
    var enrichissement = new EnrichissementPuissanceDin(actual);

    // Act
    var result = await enrichissement.EnrichirAsync();

    // Assert
    if (expected == null)
    {
        Assert.IsNull(result);
    }
    else
    {
        Assert.AreEqual(expected, result.Value);
    }
}
```

Figure 67 - Exemple de test unitaire sur l'enrichissement

Ci-contre un exemple de test unitaire sur l'enrichissement de la puissance DIN, avec différents cas de tests.

La fonctionnalité « TestCase » a été utilisée ici afin d'optimiser la réutilisation du code.

```
[Test]
public void CalcDistance_With_Different_Versions_Inverted_Must_Return_Distance_Ok()
{
    // Arrange
    var dimRef = new Dimension("A3 Berline 35 TDI", ListDimensionsVehicules.VERSION, 1);
    var dimTest = new Dimension("A3 Berline 35 TDI 150 S tronic 7",
                                                    ListDimensionsVehicules.VERSION, 1);
    var calculDistanceString = new CalculDistanceVersion(dimRef, dimTest);

    // Act
    var result = calculDistanceString.CalcDistance();

    // Assert
    Assert.AreEqual(0.5F, result);
}
```

Figure 68 - Exemple de test unitaire sur la calcul de distance

Un autre exemple de test, ci-dessus, concerne le calcul de distance dans le cas d'une chaine de caractères (qui n'est ni énergie, ni version).

Au total, plus de 550 tests unitaires ont été mis en place sur tous les aspects de l'algorithme, afin de sécuriser au maximum les futures évolutions et éviter les régressions.

5.4.2 Les tests d'intégration

Autre type de tests mis en place, les tests d'intégration permettent de tester l'ensemble de l'API en condition réelle, avec accès aux bases de données.

Ils permettent de vérifier la cohérence des résultats en environnement réel, avec appel à tous les services nécessaires à l'exception ici du SIV Argus, qui nous le rappelons est payant, et sera donc mocké.

En annexe 5.4.2, un exemple de test mis en place sur la réconciliation par immatriculation, avec un premier cas où le format d'immatriculation n'est pas correct, et l'autre où le format est correct et donc doit donner un résultat cohérent (réponse http 200).

Ces tests sont beaucoup moins nombreux car beaucoup plus long à exécuter et à mettre en place. Néanmoins nécessaires pour vérifier l'intégrité de l'entièreté du service.

5.4.3 Les tests de non-régression

Le dernier type de tests utilisé ici, les tests de non-régression, vont tester l'intégralité d'un service « morceau par morceau ». Afin de vérifier la survenue de la moindre modification dans celui-ci.

Par exemple, il peut tester l'appel à telle ou telle fonction contenue dans le service avec les paramètres attendus, comme dans l'exemple ci-dessous représentant un test du service de réconciliation, qui vérifie si la fonction de récupération des véhicules candidats est bien appelée avec les bons paramètres.

```
public class ReconcilierVehiculeServiceTests
{
    private Mock<ICatalogueConstructeurAdapter> _handlerMock;
    private IReconcilierVehiculeService _service;

    [SetUp]
    public void Setup()
    {
        _handlerMock = new Mock<ICatalogueConstructeurAdapter>();
        _service = new ReconcilierVehiculeService(_handlerMock.Object);
    }

    [Test]
    public async Task
    ReconcilierModeleVehicule_Calls_GetListVehiculeCandidatsAsync_WithCorrectParameters()
    {
        var mockVehicule = new Mock<IInfosVehicule>();
        var mockStrategy = new Mock<IStrategieReconciliationModeleVehicule>();

        _handlerMock.Setup(x => x.GetListVehiculesCandidatsAsync(mockVehicule.Object,
                                                                 mockStrategy.Object)).
            Returns(Task.FromResult(new List<InfosVehicule>()));

        var result = await _service.ReconcilierModeleVehicule(mockVehicule.Object,
                                                               mockStrategy.Object);

        _handlerMock.Verify(x => x.GetListVehiculesCandidatsAsync(mockVehicule.Object,
                                                                  mockStrategy.Object), Times.Once);
    }
}
```

Figure 69 - Exemple de test de non-régression

L'ensemble de ces tests permet la prise en main plus facile de l'application par des tiers, le tout en préservant les fonctions vitales de l'algorithme. C'est donc une partie importante du développement qui permet une meilleure maintenabilité, et des possibilités d'évolutions plus sécurisées.

5.5 Déploiements et mise en production

5.5.1 Les différents environnements

Nous avons donc maintenant une application fonctionnelle et testée. Il faut donc savoir où, et à quelles conditions l'héberger.

Comme vu précédemment, CGI Finance utilise Azure DevOps et Kubernetes pour héberger ses solutions.

Un espace nous a donc été octroyé sur Azure DevOps, avec un repository Git, et des pipelines préconfigurés pour le CICD.

En ce qui concerne les environnements, il en existe quatre chez CGI Finance :

- La DEV (ou intégration) : environnement où nous avons tout pouvoir.
- La REC (ou recette) : environnement où l'équipe de tests uniquement peut nous donner droit de déployer.
- La REC-DYN (ou recette dynamique) : environnement de recette alternatif, le déploiement sur l'un ou l'autre est décidé par l'équipe de tests en fonction des travaux en cours.
- La PROD (ou production) : environnement final sur lequel les applications sont disponibles pour les clients. Le déploiement sur cet environnement est très réglementé, et nous n'avons aucun droit dessus (juste accès en lecture seule de la base de données).

En Annexe 5.5.2, est présente une capture d'écran de l'interface d'Azure DevOps illustrant ces différents environnements.

5.5.2 La validation par l'équipe Testing

Une fois qu'une version de l'application est jugée stable, totalement fonctionnelle et prête à partir en production, une demande à l'équipe testing est faite afin de déployer en REC.

Une fois validé et déployé, c'est alors que l'équipe entre en jeu pour effectuer des tests sur l'application.

Des tests de non-régression par rapport à la version précédente (ou par rapport à l'application précédente dans le cas du premier déploiement), qui servent à vérifier que les résultats sont cohérents avec ceux attendus, et se prémunir contre une dégradation dans la qualité de service.

Des tests de charge pour vérifier que l'application, en simulant de nombreux utilisateurs simultanés, reste stable et garde un temps de réponse raisonnable.

La communication se fait par JIRA suivant les différents tests, et les éventuelles modifications nécessaires se font rapidement. C'est une période où il faut être très réactif car l'équipe de test n'est mobilisée qu'un temps donné par application, et le respect de ce délai est totalement indispensable.

C'est à ce moment que nous avons dû effectuer des modifications, notamment sur la requête de récupération des véhicules candidats qui était longue à s'exécuter, ou sur l'ajout de différents index sur la base de données, afin d'améliorer drastiquement les performances.

Une fois les tests validés, l'équipe testing rédige un PV de recette qui sera nécessaire pour l'étape suivante : le passage devant le C.A.B.

5.5.3 Le Change Advisory Board (C.A.B.)

Le Change Advisory Board (C.A.B.) est une instance au sein de CGI Finance qui va valider ou on les passages en production des nombreuses applications.

Elle est composée de différents chefs de services techniques et des architectes, qui vont analyser les impacts qu'une mise en production pourrait avoir sur l'ensemble du S.I., vérifier si toutes les étapes nécessaires ont été respectées (comme la fourniture du PV de recette), déterminer les impacts sur les éventuels clients, et également de garantir la traçabilité de toutes les mises en production.

Elle se déroule en réunion ou chaque demandeur présente le projet, les évolutions faites, et les risques qui pourraient se présenter suite à une mise en production.

Dans notre cas, l'application n'ayant encore jamais été mise en production et n'ayant encore aucun client la consommant, l'impact sur le S.I. était donc nul. La seule véritable exigence fut d'avoir le PV de recette.

Le fait de mettre en production avant que les clients ne se branchent sur l'application nous a permis de vérifier si toutes les fonctionnalités, sur ce nouvel environnement, fonctionnaient bien comme attendu.

Plus tard, d'autres C.A.B. ont été organisés, mais du coté des clients (Occazio et Téléfi), afin de se brancher sur notre service, et ce une fois qu'ils ont été prêt à cela.

Le C.A.B. est donc le garde-fou de l'intégrité de la production, qui est l'environnement critique car tout souci sur celui-ci impactera les clients, et à fortiori les clients finaux (les utilisateurs d'Occazio et les concessionnaires dans le cas de Téléfi).

5.5.4 Les déploiements

Et enfin, sur la partie technique des déploiements (valable pour tous les environnements), des pipelines CICD ont été mis en place. Ainsi, à partir du repository présent sur Azure DevOps, le déploiement en lui-même comporte les étapes suivantes :

- Ajout d'une étiquette de version à l'application. (Annexe 5.5.4.a).
- Lancement du pipeline de build de l'application. (Annexe 5.5.4.b). Dont on voit le détail à l'annexe 5.5.4.c avec le résultat d'une partie de l'exécution des tests.
- Lancement du pipeline de déploiement. (Annexe 5.5.4.d). C'est à cette étape que l'on sélectionne l'environnement où déployer.
- Enfin le lancement du pipeline de déploiement du package Nuget. (Annexe 5.5.4.e). Ce package Nuget comprend le projet Message qui contient tous les objets d'échange de l'application. Ainsi les clients peuvent l'utiliser pour être sûr d'avoir les objets à jour pour l'utilisation de l'API.

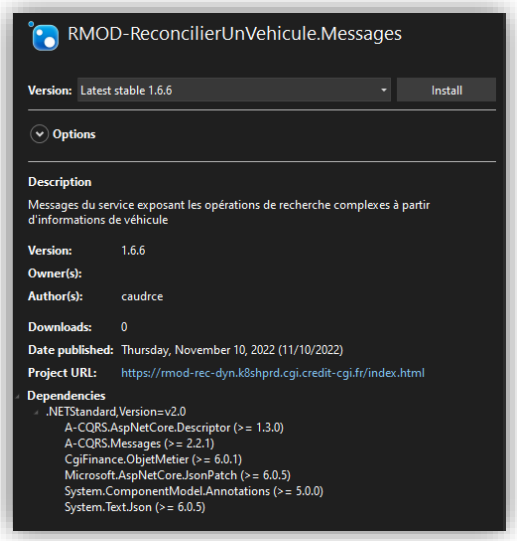


Figure 70 - Package Nuget mis à disposition sur le repository interne de CGI Finance

5.6 Conclusion sur le développement

L'application est maintenant déployée en production et prête à être consommée par les clients. C'est le début d'une nouvelle phase qui va devoir être menée en parallèle des améliorations : le run. En effet, une fois en place, les différents problèmes vont commencer à survenir, et il faut pour cela mettre en place des outils de supervision afin que cela se passe pour le mieux, ce qui sera l'objet de la partie suivante.

6 EXPLOITATION

6.1 Mise en place de HealthChecks

6.1.1 HealthCheck sur RMOD

Afin d'assurer un suivi efficace de la santé du service, un endpoint spécial, appelés healthcheck, a été mis en place.

Ce endpoint ne déclenche aucune action spécifique, mais renvoie simplement une réponse signifiant de l'état de santé du service.

Ce healthcheck a été implémenté sur la base de données, vérifiant si elle est toujours accessible. Ainsi, si tel est le cas, une réponse http 200 est renvoyée. Tout autre réponse sera le signe d'un problème avec la base.

Il sera utilisé par Datadog afin de faire de la supervision en temps réel de l'état de santé de RMOD.

6.1.2 HealthCheck sur le SIV Argus

Le SIV Argus, comme vu à de nombreuses reprises, est un service externe payant géré par l'Argus. Il n'avait pas de Healthcheck disponible. Or, afin de pouvoir établir un périmètre précis en cas de problème sur RMOD, il était intéressant d'en avoir un pour exclure une éventuelle défaillance de ce service.

De plus, il n'était pas envisageable de faire des requêtes vers le SIV Argus juste pour vérifier son état de santé, car nous rappelons que ce service est payant et cela aurait entrainé une hausse importante des coûts.

Une réunion avec des commerciaux de l'argus a donc été organisée, où nous avons demandé la faisabilité de l'implémentation d'un tel endpoint sur leur service.

La réponse fut positive et il fut mis en place quelques semaines plus tard.

Ainsi, comme pour RMOD, nous pouvons donc s'assurer de la bonne santé du service externe via un endpoint, gratuit cette fois, avec de la supervision dans Datadog, outil que nous allons aborder maintenant.

6.2 Datadog

6.1.1 Présentation

Datadog est une plateforme de monitoring et d'analytique pour les environnements cloud. Elle joue un rôle essentiel dans l'exploitation des applications en fournissant une visibilité en temps réel sur les performances des serveurs, des conteneurs, des bases de données et des services. Dans le cadre de RMOD, l'utilisation de Datadog est prévue pour plusieurs raisons clés :

Figure 71 - Logo de Datadog

- **Surveillance en Temps Réel :** Datadog permet de surveiller les performances de l'application, détectant les problèmes tels que les temps de réponse lents, les erreurs de serveur, et les goulots d'étranglement de ressources.
- **Alertes et Notifications :** Elle offre des fonctionnalités de notification en temps réel en cas de détection de problèmes, permettant une intervention rapide pour maintenir la stabilité de l'application.

6.1.2 Surveillance en temps réel

Datadog permet donc de faire de la surveillance d'applications en temps réel. Dans notre cas, il permet d'avoir une visualisation de toutes les requêtes effectuées vers et par RMOD, ainsi que leurs statuts. Ceci permet donc de surveiller chaque requête individuellement si besoin, et de voir quels composant de l'application peut rencontrer des problèmes.

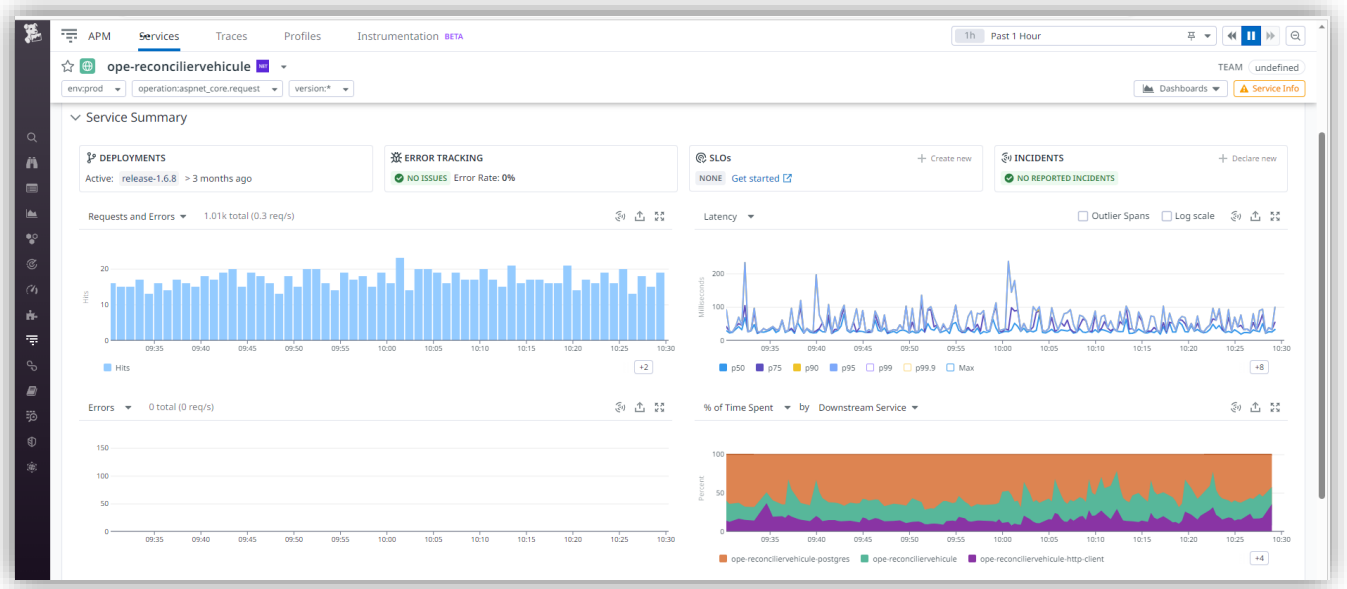


Figure 72 - Dashboard de Datadog

Il permet d'avoir également des détails sur chaque requête, des appelants, et des services qu'il appelle ainsi que dans quel volume. Comme le montre la figure 74.

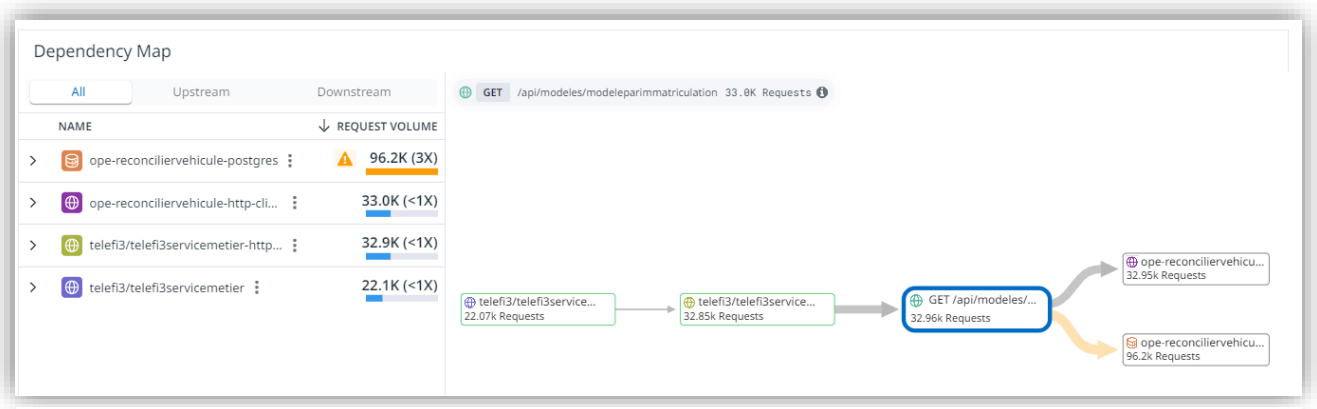


Figure 73 - Détail de requête dans Datadog

6.1.3 Alertes et notifications

Datadog est également capable de faire des requêtes vers les applications, et de créer des alertes suivant des conditions.

C'est ici qu'entrent en œuvre les healthchecks abordés plus tôt.

Une liste de tests a donc été mise en place, afin de vérifier les différents healthchecks, que ce soient ceux de RMOD avec la disponibilité de la base de données, la disponibilité elle-même de l'application (en allant vérifier la disponibilité du lien /index.html), ou même du SIV Argus.

Les tests sont configurés sur chaque environnement, hormis bien entendu le SIV Argus sur lequel nous n'avons pas la main sur les autres environnements que la PROD. (Annexe 6.1.3).

De plus, une notification dans un groupe Teams est envoyé dès lors qu'un test n'a pas été concluant. Ceci a été mis en place car Teams est toujours activé sur le poste, mais pas forcément Datadog, donc cela favorise une meilleure réactivité.

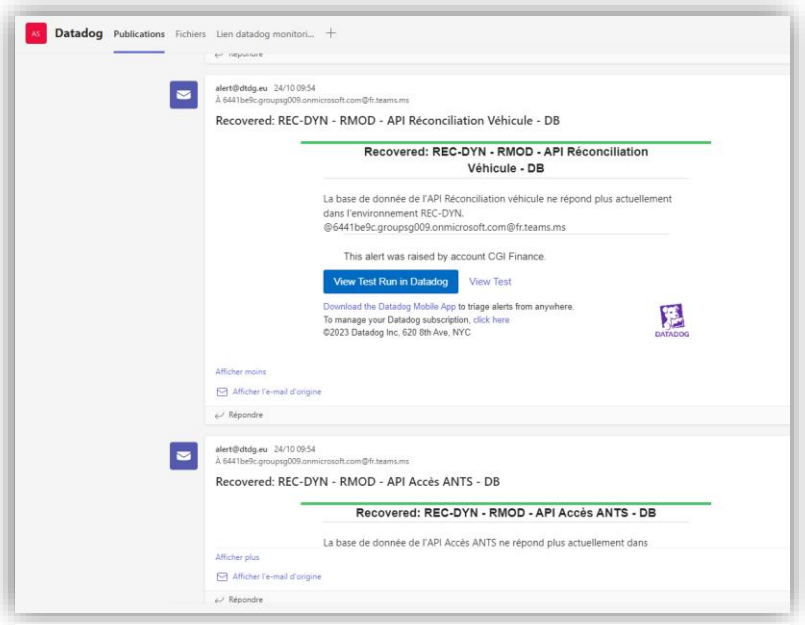


Figure 74 - Alerte Datadog dans Teams

La configuration des tests et des alertes est de plus très simple, avec pour exemple ci-dessous le test de disponibilité de RMOD en PROD.

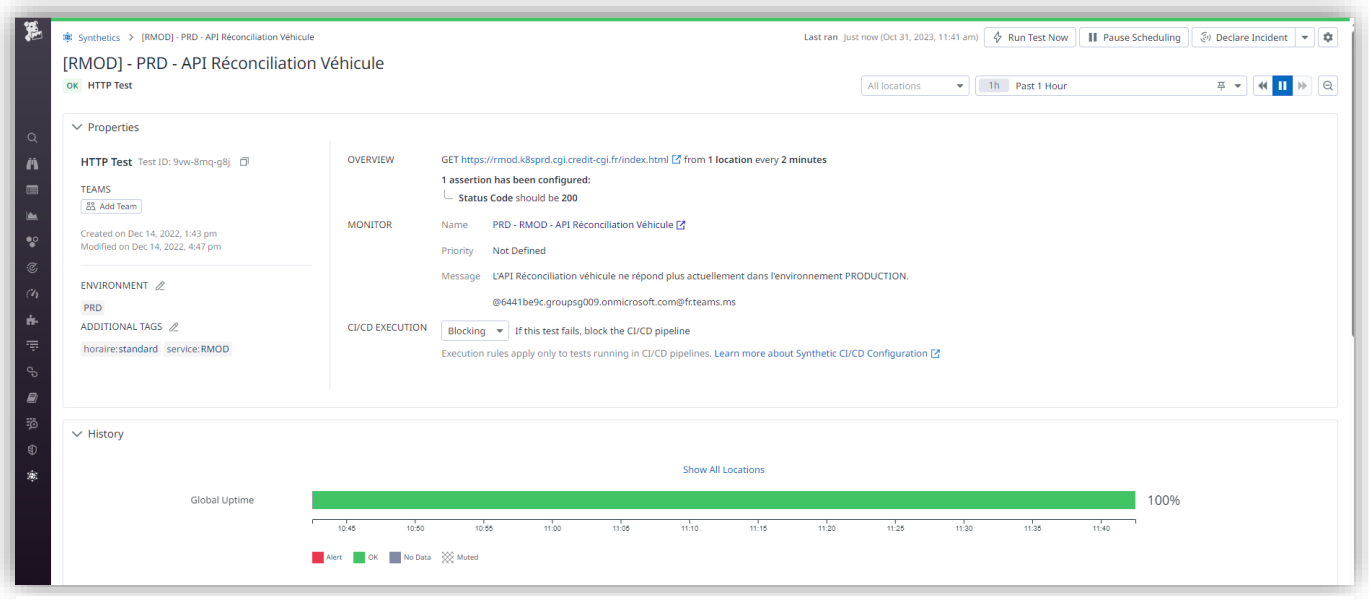


Figure 75 - Configuration du test dans Datadog

Datadog est donc un outil complet, intuitif afin de suivre la santé de l'application. Pour aller un peu plus en profondeur en cas de problème, on peut faire une analyse des logs, et ceci sera fait avec ELK dont nous allons aborder le principe maintenant.

6.3 ELK

6.3.1 Présentation

ELK, un acronyme pour Elasticsearch, Logstash et Kibana, est une suite puissante et populaire pour la gestion des logs et l'analyse des données. Dans le cadre de RMOD, l'utilisation d'ELK présente plusieurs avantages significatifs :

Figure 76 - Logos d'ELK

- **Elasticsearch :** Une moteur de recherche et d'analyse distribué, Elasticsearch stocke et analyse les grandes quantités de données générées par l'application. Il est extrêmement rapide, permettant des recherches quasi instantanées, même dans de grands volumes de données.
- **Logstash :** Un outil de traitement de logs, Logstash collecte, transforme et achemine les données vers Elasticsearch. Il joue un rôle essentiel dans la normalisation des logs issus de différentes sources.

- **Kibana :** Une interface utilisateur pour visualiser les données stockées dans Elasticsearch. Kibana permet de créer des tableaux de bord personnalisés pour une analyse approfondie des logs, offrant une vue d'ensemble claire de l'activité de l'application et facilitant la détection rapide des problèmes.

6.3.2 Utilisation

Le framework CGI a été conçu pour prendre en charge nativement la gestion des logs par ELK. Il suffit d'une petite configuration pour cela, et nous avons ensuite accès aux logs dans chaque environnement.

En Annexe 6.3.2, un exemple de l'interface, avec l'affichage d'un log de l'application.

C'est très utile pour naviguer rapidement parmi les milliers de logs générés en une journée, et de détecter les différents problèmes qui ont pu survenir.

6.4 Gestion des incidents

Le dernier aspect du RUN auquel nous avons été confronté est la gestion des incidents après la mise en production de RMOD.

Dans notre cas, ce fut surtout de la recherche d'explication lorsque les résultats de l'application étaient incohérent ou innatendus.

Le process dans ce cas est le suivant (dans Téléfi en particulier) :

- Le client final fait appel au HelpDesk quand il detecte un problème.
- Le HelpDesk fait remonter l'information à l'équipe technique.
- Une recherche des responsabilité est effectuée : un ticket JIRA est créé.
- Le ticket nous est attribué quand la délimitation de l'incident est précisée autour de RMOD.
- Nous cherchons la cause de l'incident.
- Nous répondons sur le ticket JIRA avec la correction si elle est possible de suite sans développement supplémentaire, ou mis à la liste des tâches pour la version ultérieure sinon.

Les incidents ont été plutôt rares, surtout en comparaison au volume d'appel. Et ceux existant nous donnent la plupart du temps des pistes pour améliorer l'algorithme, après analyse des données, ce qui sera traité dans la partie suivante.

6.5 Rédaction de la documentation utilisateur

En parallèle, une documentation utilisateur a été rédigée par nos soins, afin d'expliquer aux clients comment utiliser RMOD.

Elle comprend donc toutes les informations sur les endpoints, les objets à utiliser, où trouver le package Nuget Messages, un arbre de décisions (Annexe 6.5), et également quelques parcours types afin de mener à bien une réconciliation.

Par exemple ci-dessous le parcours type avec retranscription marque et modèle.

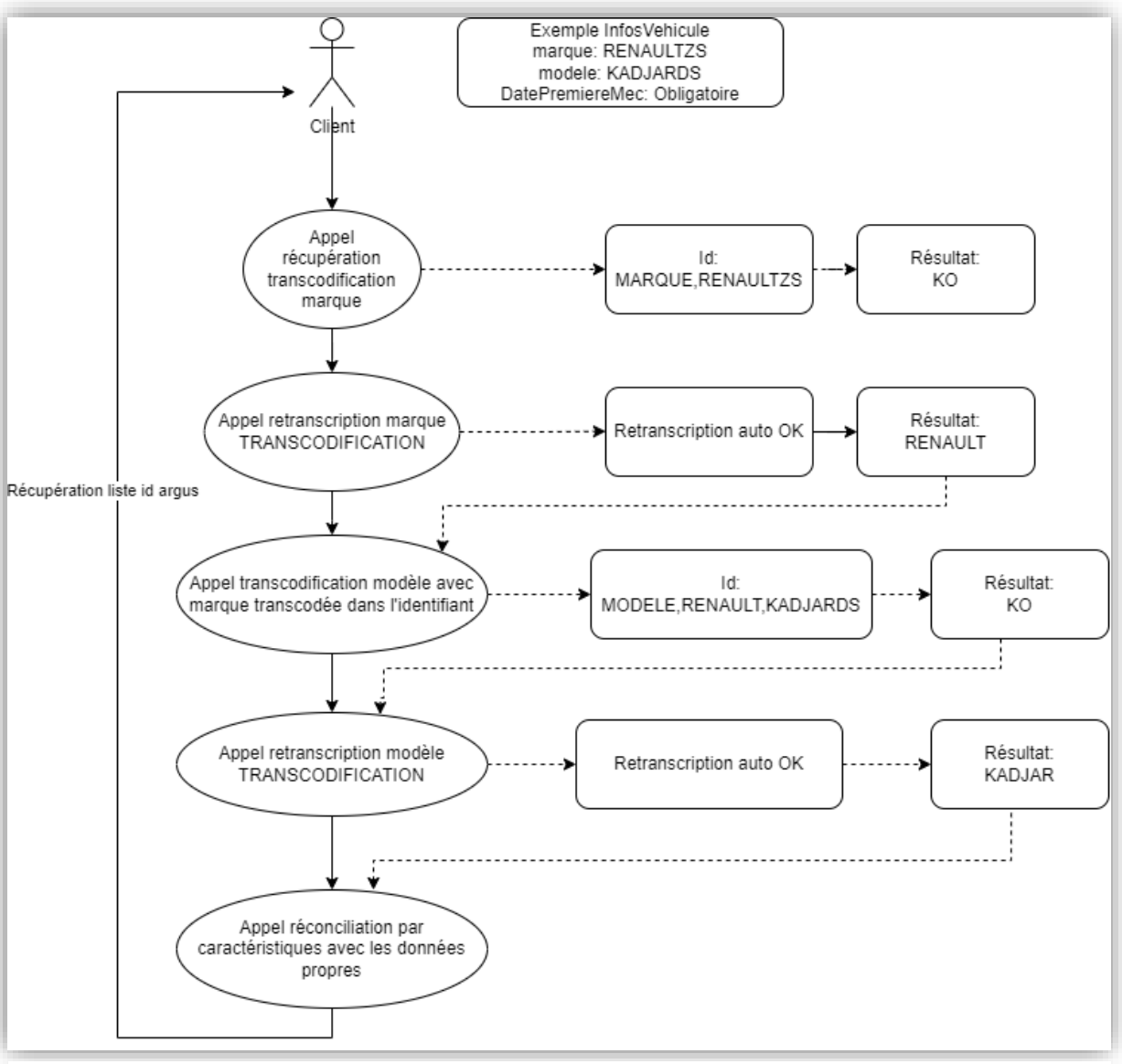


Figure 77 - Parcours type d'utilisation de RMOD

7 ANALYSE ET VALIDATION

7.1 Objectifs

Figure 78 - Logo WPF

Le projet RMOD a une forte composante data, et sa conception a nécessité de pouvoir trouver des stratégies de réconciliations donnant les meilleurs résultats possible.

Ceci a donc necessité le développement d'un outil d'automatisation des appels à partir de fichier csv, qui a pris la forme d'une application lourde en .NET WPF, et l'utilisation d'un outil de Dataviz, PowerBI.

Ces analyses ont été le sujet de la plupart des réunions, intervenant environ toutes les deux semaines, avec le chef de service et le tech lead. De très nombreux ajustements ont été fait au fur et à mesure, autant dans le test de différentes stratégies, que dans l'analyse des problèmes et les évolutions algorithmiques qui en ont découlées.

Il sera donc totalement impossible de présenter l'intégralité du travail d'analyse qui a pu être effectué dans le cadre de ce mémoire, mais nous allons voir comment les outils ont été mis en place, et montrer un exemple d'analyse sur un fichier donné.

7.2 L'application lourde

7.2.1 Architecture

L'application lourde développée, utilisant le framework WPF inclus dans .NET, utilise l'architecture MVVM (Model – View – ViewModel) afin de séparer les responsabilités de ses composants.

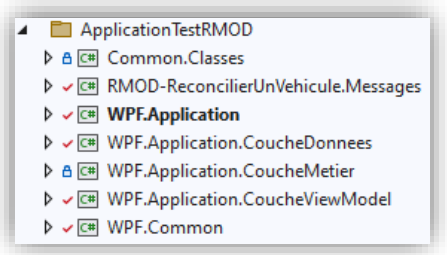


Figure 79 - Hiérarchie des projets de l'application lourde

Nous n'entrerons pas dans le détail du développement de l'application en elle-même car ce serait superflu, l'essentiel pour notre sujet est de comprendre à quoi elle sert et quelles sont ses capacités, ce qui sera vu dans la partie suivante.

7.2.2 Principe

L'application a pour but d'automatiser les appels à RMOD à partir de fichier csv contenant des caractéristiques de véhicules, tel que ceux vus aux Annexes 4.2.1.a et 4.2.1.b, afin d'enregistrer les résultats dans un format adapté à l'analyse PowerBI. De plus, elle doit être capable et c'est tout son intérêt, de lancer les requêtes avec différentes stratégies pour une même analyse.

Le but est également que l'application soit assez simple d'utilisation, et qu'elle ne requiert pour l'utiliser, pas ou peu de compétence en développement. C'est pour cette raison que nous avons choisi une application lourde.

Compte tenu de ces exigences, deux problèmes principaux sont identifiés :

- Les fichiers csv ne sont pas toujours formattés de la même façon, et les colonnes ne sont quasiment jamais dans le même ordre suivant les sources. Il faut donc y implémenter un système de mapping des colonnes vers les caractéristiques que l'on veut y extraire, avec un système de persistance.
- Il faut un endroit où stocker les résultats qui soit facilement récupérable par PowerBI et ceci de manière intuitive. Le choix de créer un modèle dans PostgreSQL, avec un nouveau schéma pour chaque analyse, a donc été retenu. Un fichier PowerBI sera donc créé et pour choisir telle ou telle analyse il ne suffira que de changer le schéma dans la source de données.

7.2.3 Interface

L'interface de l'application se divise donc en deux fenêtres principales, accessibles via un menu. Concernant les captures, nous sommes désolé de ne pas pouvoir en montrer avec l'application fonctionnelle, et ce pour une raison dans laquelle nous reviendrons dans la conclusion, mais nous mettons celles du concepteur graphique de Visual Studio…

La première, concernant la configuration du mapping (Annexe 7.2.3.a) se présente en quatre grandes parties :

- La première avec la sélection du fichier et affichage des données avec un numéro de colonne.
- La deuxième avec les différentes caractéristiques utilisables, dans lesquels on reporte le numéro de la colonne correspondante, et où on lui donne un nom.
- La troisième pour valider la configuration.
- La quatrième qui montre les configurations déjà existantes.

La deuxième fenêtre, elle, concerne l'interface de lancement et de visualisation de l'avancée des requêtes en fonction du nombre de lignes dans le fichier sélectionné (Annexe 7.2.3.b).

Elle se divise en ces parties suivantes :

- Le choix de l'environnement (LOCAL et DEV), les autres environnements n'ont pas été implémentés par manque de droit, et surtout car les tests en LOCAL, en récupérant toutes les données à jour, se révélaient plus performants sans perte de qualité, le tout en donnant la possibilité de faire des modifications d'algorithme rapide si besoin.
- Le choix du fichier.
- Le choix de la configuration de mapping correspondant au fichier.
- Le choix des différentes stratégies à tester.
- L'ajout des informations (non obligatoires) du nom d'utilisateur et du service.
- Le bouton de lancement.
- Un récapitulatif des informations du fichier.
- Un aperçu brut des premières lignes du fichier.
- La liste des stratégies sélectionnées et leurs détails.
- Et enfin la barre de progression.

7.2.4 Le modèle de données des résultats

Les résultats sont donc enregistrés dans la base de données PostgreSQL. A chaque analyse correspondant un nouveau schéma de base.

Le modèle de données a été conçu pour être le plus exhaustif possible, en récupérant pour chaque véhicule toutes les informations sur chaque requête effectuée, que ce soit un appel pour récupérer une transcodification, pour un enrichissement ou une transcodification automatique, ainsi que, pour chaque stratégie utilisée, les résultats de la requête de réconciliation par caractéristiques.

De plus, dans le cas où nous avons la plaque d'immatriculation, l'application enrichi les données brutes du véhicule avec les informations venant du SIV, dans le cas où elles seraient manquantes, ce qui constituera notre table de faits : les infos véhicules enrichis.

Le modèle est de type étoile, et prêt à être consommé par PowerBI, que nous allons aborder dans la partie suivante.

Le schéma du modèle est en Annexe 7.2.4.

7.3 Analyse des données dans PowerBI

7.3.1 Présentation

Figure 80 - Logo PowerBI

PowerBI est un service d'analyse business de Microsoft, offrant des fonctionnalités interactives de visualisation de données et de business intelligence. Cet outil permet de transformer de grandes quantités de données brutes en rapports et dashboards visuels informatifs, facilitant la prise de décision basée sur des données.

Dans le cadre de RMOD, il va permettre, en consommant le modèle de données précédemment décrit, de créer des visuels permettant de juger de la qualité de la réconciliation, ainsi que de la mise en qualité des données.

7.3.2 Précision sur les données utilisées dans cet exemple

Dans l'exemple d'analyse que nous allons voir par la suite, il s'agit de véhicules provenant du partenaire PVO (Planet VO).

Utiliser les données de ce partenaire est pertinent dans le cadre d'évaluer la performance de RMOD car il a, pour la plupart des véhicules, déjà un id argus renseigné. De plus, les experts métiers considère cet id argus comme fiable, et donc il pourra servir de base de comparaison pour les performances de RMOD.

A noter que dans beaucoup de cas avec d'autres partenaires, certains visuels sont inexploitables, car n'ayant pas cet id argus comme base de comparaison renseigné.

Nous allons maintenant passer à l'analyse de quelques visuels. Ce ne sera pas exhaustif car il y en a beaucoup trop (26), mais nous allons en sélectionner quelques-uns qui sont les plus représentatifs de l'apport de PowerBI dans l'analyse des résultats de RMOD.

7.3.3 Visuel d'analyse générale

Le premier visuel (Annexe 7.3.3), regroupe des informations très générales sur l'analyse effectuée sur le fichier.

Il comprend le nombre de véhicule ayant fait l'objet de requêtes à RMOD, et le nombre de sources d'où elles proviennent (ici une seule, PVO, il peut y avoir des fichiers avec plusieurs sources).

Il montre également le nombre de stratégies utilisées sur ce fichier.

Et enfin propose une première information sur les marques et modèles, entre le nombre présents initialement (donnée brute), et le nombre après mise en qualité (après transcodification).

7.3.4 Visuel d'analyse de la date de mise en circulation

Dans ce visuel (Annexe 7.3.4), nous pouvons analyser la répartition des véhicules par date de mise en circulation présentes dans le fichier.

On peut déjà déceler quelques problèmes :

- 6.57% des véhicules n'ont pas de date MEC renseignée, les rendant inéligibles à une réconciliation par RMOD.
- Il existe des dates aberrantes, comme la plus ancienne étant au 1er janvier 1900.
- La répartition nous montre que, malgré que la grande majorité des véhicules sont compris entre 2010 et 2023, un nombre non négligeable a une date MEC de plus de 10 ans (qui est la limite d'âges des véhicules d'occasion financés par CGI Finance).

Suite à ces observations, des filtres seront appliqués par la suite afin de juger la performance de RMOD uniquement sur les véhicules potentiellement réconciliables, car garder ces résultats biaiserais la performance globale de l'algorithme, l'important ici étant de documenter ces cas où la réconciliation ne peut aboutir.

7.3.5 Visuel d'analyse de l'enrichissement sous-modèle

Ce visuel (Annexe 7.3.5), permet de juger de la performance de RMOD en ce qui concerne l'enrichissement des données, en particulier sur les sous-modèles.

Dans ce cas, nous pouvons constater la différence de véhicules avec le sous-modèle enrichi avant et après l'enrichissement. Nous constatons donc que sur l'échantillon complet de véhicule, 16.93% ont eu un enrichissement réussi.

Le visuel, donc, mesure la performance de l'enrichissement sous-modèle, et, via les tableaux montrant la granularité la plus fine, permet de vérifier visuellement s'il y a des erreurs ou non dans cet enrichissement. Ce qui permet, en cas de souci, de corriger l'algorithme afin d'éviter les erreurs.

7.3.6 Visuel d'analyse de l'enrichissement finition

Ce visuel (Annexe 7.3.6), permet, de la même façon que l'enrichissement sous-modèle, permet de juger de la performance de l'enrichissement de la finition à partir du champ version.

La différence avec le précédent est :

- Qu'il montre également les véhicules qui ont la finition initiale vide (100% ici) et ceux qui n'ont pas la version renseignés (0% ici). Ces derniers n'étant pas éligibles à l'enrichissement.

- Il montre de plus, dans le tableau au niveau de granularité le plus fin, la correspondance avec la version. Ainsi, nous pouvons voir les cas dans lesquels l'enrichissement a réussi, et ceux ou, il aurait dû fonctionner mais sans succès. Et ainsi donner des pistes pour l'investigation afin d'améliorer encore une fois l'algorithme.

Ce visuel est donc une représentation de la performance de l'enrichissement de la finition à partir du champ version par RMOD, et permet de visualiser les éventuels problèmes dans la vue d'une évolution future pour les corriger.

7.3.7 Visuel d'analyse générale de la réconciliation

Passons maintenant à quelques visuels concernant la réconciliation. Dans celui-ci (Annexe 7.3.7), nous pouvons observer les résultats généraux de la réconciliation sur le fichier.

Première chose à noter est le segment « Date MEC valable », qui représente, comme vu précédemment, les véhicules dont la date MEC est comprise dans l'intervalle où le véhicule est potentiellement réconciliable.

Les deux graphiques représentent le pourcentage de réussite de réconciliation, le premier en utilisant le statu de réconciliation (réussite ou échec) général, et le second avec le détail de réconciliation, deux attributs qui font partie de la réponse de RMOD comme vu dans la partie conception.

Ainsi, avec le visuel de droite, en ajoutant le pourcentage de réussite des statuts UN_SEUL_MODELE (72.67%) et PLUSIEURS_MODELE_AVEC_PRIX_INFERIEUR_A (16.26%), nous retrouvons bien le pourcentage de succès du visuel de droite (88.93%).

Chose importante à noter : ce taux de réussite est celui proposé par RMOD. C'est-à-dire lorsqu'il a trouvé au moins un résultat, et quand la différence de prix des id argus renvoyé est inférieure au pourcentage défini dans la stratégie. Cela ne dit pas si ce sont réellement les bons véhicules, ce que nous allons essayer de discerner dans le dernier visuel que nous aborderons.

7.3.8 Visuel d'analyse de la réconciliation en comparaison avec l'id argus cible

Enfin, le dernier visuel que nous allons aborder, est celui qui compare les résultats de RMOD par rapport aux id argus déjà renseignés dans le fichier, donc ici provenant de PVO, réputés corrects et donc appelé id cible.

Deux nouveaux KPIs ont été ici utilisés :

- Le taux de succès validé : le taux de véhicules dont le premier résultat de RMOD correspond à l'id cible.

- Le taux de succès presque validé : le taux de véhicules dont le l'id cible fait partie de la liste de résultats de RMOD, mais pas en première place.

Un nouveau segment fait également son apparition ici, étant si l'id cible est bien présent dans le catalogue constructeur. En effet, il a été constaté que nombre d'id donnés par PVO étaient absent de notre version du catalogue constructeur, ainsi, nous étions naturellement dans l'impossibilité de le retrouver avec RMOD. Cela arrive, pour PVO, dans le cas de véhicules trop anciens par rapport à notre base. Ainsi, en appliquant le segment de la date MEC valable, tous les véhicules de ce fichier ont bien leur id argus cible dans le catalogue constructeur. Néanmoins, il était intéressant de garder la distinction sur d'autres fichiers moins propres, où l'id argus proposé pouvait ne pas être dans le catalogue constructeur malgré une date de MEC valable.

Ces deux derniers indicateurs sont donc ceux qui vont mesurer la précision effective de RMOD, et, en variant les pondération, le but est de trouver celles qui donnent le meilleur taux de résultat succès validés + succès presque validés.

Ici, le taux de succès est donc de 76% + 4% = 80%. On peut donc considérer que RMOD a eu raison dans 80% des cas sur le fichier étudié. Ce qui est un score acceptable pour le métier.

7.3.9 Conclusion sur l'analyse

Nous avons donc présenté ici un seul cas, sur un fichier relativement propre, qui comprend un id argus proposé pour la plupart des véhicules et qui est réputé correct, ce qui n'est malheureusement pas toujours le cas…

Ainsi, il a fallu de nombreuses analyses, sur différents fichiers à qualité variables, afin d'affiner les pondérations à utiliser sur les différentes sources de données.

De plus, nous n'avons présenté qu'un nombre limité de visuels provenant du fichier PowerBI, mais il s'agissait des plus représentatifs pour l'analyse de ce fichier. Pour d'autres, ce seront d'autres visuels qui peuvent montrer les éventuels problèmes, comme des transcodifications manquantes, des soucis de données manquantes etc… Le rapport a été conçu pour être le plus générique possible, et de couvrir tous les aspects potentiellement problématiques de la réconciliation.

8 CONCLUSION

8.1 Etat actuel

Le projet RMOD a atteint un stade significatif : il est désormais en production en version 2, et est utilisé par trois clients majeurs, à savoir Occazio, Téléfi 3 et 4. Actuellement, ces clients utilisent RMOD principalement pour la réconciliation par immatriculation. Un déploiement élargi est cependant en perspective, notamment pour la mise en qualité et la réconciliation du stock de véhicules provenant de Vivafi via la réconciliation par caractéristiques.

Un aspect notable du développement de RMOD est la mise en évidence des problèmes persistants liés à la qualité des données, aussi bien en provenance des systèmes de gestion de base de données (DMS) qu'en interne. Cette prise de conscience a catalysé la nécessité d'adopter des mesures proactives pour améliorer de manière systématique la qualité des données au sein du Système d'Information (S.I.). Ainsi, plutôt que de se limiter à "réparer les dégâts", une démarche proactive est, normalement, envisagée pour traiter ces défis à la source lorsque cela sera possible.

8.2 Bilan

Le bilan du projet est donc globalement positif et a répondu à la plupart des besoins métiers initialement évoqués, même s'il reste encore loin d'être parfait.

De plus, il propose maintenant des solutions inédites, et totalement indépendantes, au sein du S.I. concernant la mise en qualité des données, en attendant les mesures proactives. Il permettra une mise en qualité, certes incomplète, mais bien meilleure que l'existant avant RMOD de par sa généricité et sa modularité.

Au niveau chiffres, RMOD aura permis de réduire de façon très significative la facture des appels au SIV Argus. D'environ 60.000 euros mensuels, nous sommes passés à environ 15.000 euros, ce qui révèle de nettes économies, même si ces chiffres sont encore perfectibles.

D'un point de vue personnel, RMOD nous aura permis de découvrir le S.I. et l'organisation d'une grande entreprise, avec ses avantages et ses inconvénients.

Les avantages auront été la grande concentration d'acteurs de talents, auprès de qui nous avons pu apprendre énormément.

Les inconvénients étant la grande rigidité de ce genre de S.I., notamment (mais c'est compréhensible), au niveau de la sécurité, ce qui nous aura posé quelques soucis qui seront abordés dans la partie suivante.

8.3 Les principales difficultés

Les principales difficultés rencontrées auront été au niveau de l'explosion des responsabilités entre les services. En effet, il était difficile, au sein d'un S.I. de cette ampleur, d'avoir les réponses adéquates à nos interrogations, et ce dans un délai assez court.

De plus, certaines rigidités inhérentes à cette organisation aura provoqué quelques tensions. Comme un léger « conflit » avec le service DBA gérant la base de données lorsque du jour au lendemain elle n'était plus accessible suite à une migration, ou sur le point de vue technique, évoqué précédemment, concernant la retranscription qui aurait dû être selon nous divisé en deux parties distinctes.

Mais la principale difficulté aura été inhérente à la sécurité que l'on pourrait croire disproportionnée. C'est compréhensible qu'un établissement bancaire essaie de se prémunir contre toute fuite de données, en bloquant énormément de sites Web ou de téléchargement d'exécutables… Ceci n'est pas le problème. Le problème est que pour l'installation de certains outils indispensables, dont Visual Studio, il faille passer par des moyens détournés et pas toujours bien documentés au sein du S.I… De plus, pour des outils qui n'ont pas été prévus par la sécurité (comme PowerBI par exemple), nous avons dû passer par des moyens détournés pour réussir l'installation… Choses assez contraignantes mais encore compréhensibles…

Mais là où nous ne sommes pas d'accord, c'est qu'alors que la rédaction de ce mémoire en était qu'à ses balbutiements, je n'ai pas pu sortir les données nécessaires à sa rédaction… Et pire, le contrat se terminant durant cette période, et même en ayant anticipé, nous n'avons pas pu sortir toutes les données dont nous avions besoin pour une rédaction sereine de ce mémoire… Nous nous sommes fait alpaguer plusieurs fois par la sécurité d'ailleurs, pour enfin être totalement bloqué malgré nos explications… Ce qui fera que nous ne sommes pas satisfait de la structure même de ce mémoire, et que nous avons eu énormément de difficulté dans sa rédaction. Ceci explique donc le manque de documentations sur certains aspects, comme la base de données, JIRA, les retours de RMOD etc…

8.4 Conclusion personnelle

Cette expérience aura donc été pour moi, aussi éprouvante qu'elle fut par moment, très bénéfique. Autant techniquement qu'humainement. Je suis néanmoins soulagé d'en être arrivé au bout. Elle m'aura ouvert les yeux sur ce qu'était réellement le travail de développeur, avec une vision plus large sur l'ensemble des tenants et aboutissants, et non être la tête dans le code du matin au soir.

Ce qui, je crois, est le véritable rôle d'un ingénieur…

Bibliographie

[1] Groupe Société Générale, «Histoire Groupe Société Générale,» [En ligne]. Available: https://www.societegenerale.com/fr/le-groupe-societe-generale/identite/histoire.

[2] Groupe Société Générale, «Présentation Groupe Société Générale,» [En ligne]. Available: https://www.societegenerale.com/fr/le-groupe-societe-generale/identite/presentation.

[3] CGI Finance, «Notre entreprise,» [En ligne]. Available: https://www.cgifinance.fr/groupe-societe-generale-innovantes/.

[4] DSquad, «Occazio,» [En ligne]. Available: https://www.occazio.fr/.

[5] Tchek, «CGI FINANCE et Tchek annoncent leur partenariat,» [En ligne]. Available: https://fr.tchek.ai/press-release/cgi-finance-et-tchek-annoncent-leur-partenariat.

[6] CGI Finance, «Innovations - CGI Finance - Auto,» [En ligne]. Available: https://www.cgifinance.fr/auto/solutions-digitalise-valeur-environnement/.

[7] Argus, «Histoire de l'Argus,» [En ligne]. Available: https://www.largus.fr/cote/histoire/argus/.

[8] Argus, «Referentiel_Argus.pdf,» [En ligne]. Available: https://www.largus.fr/pro/includes/pdf/produits/Referentiel_Argus.pdf.

[9] Institut National de la Consommation, «Le point sur l'affaire Volkswagen en 15 questions,» 25 12 2015. [En ligne]. Available: https://www.inc-conso.fr/content/le-point-sur-laffaire-volkswagen-en-15-questions.

[10] ANTS, «ANTS - Le système d'immatriculation d'un véhicule (SIV),» [En ligne]. Available: https://immatriculation.ants.gouv.fr/tout-savoir/le-systeme-d-immatriculation-d-un-vehicule-siv.

[11] L'Argus, «API Argus - Documentation,» [En ligne]. Available: https://developer.largus.fr/#28323bfe-9cd9-4b2f-9c87-9a67a0e86037.

[12] S. Oloruntoba, «SOLID : les 5 premiers principes de conception orientée objet,» [En ligne]. Available: https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design-fr.

[13] Atlassian, «Jira | Logiciel de suivi des tickets et des projets | Atlassian,» [En ligne]. Available: https://www.atlassian.com/fr/software/jira.

[14] Microsoft, «Nouveautés de .NET 6,» [En ligne]. Available: https://learn.microsoft.com/fr-fr/dotnet/core/whats-new/dotnet-6.

[15] Microsoft, «Vue d'ensemble d'ASP.NET Core,» [En ligne]. Available: https://learn.microsoft.com/fr-fr/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0.

[16] Microsoft, «Nouveautés de C# 10,» [En ligne]. Available: https://learn.microsoft.com/fr-fr/dotnet/csharp/whats-new/csharp-10.

[17] Microsoft, «Nouveautés de Visual Studio 2022,» [En ligne]. Available: https://visualstudio.microsoft.com/fr/vs/whatsnew/.

[18] Integrative Systems, «Banking and Finance Industry Needs Dot Net Core Development,» [En ligne]. Available: https://www.integrativesystems.com/banking-and-finance-industry-needs-dot-net-core-development/.

[19] Refactoring Guru, «Mediator,» [En ligne]. Available: https://refactoring.guru/design-patterns/mediator.

[20] Sonar, «Clean Code Tools for Writing Clear, Readable & Understandable Secure Quality Code,» [En ligne]. Available: https://www.sonarsource.com/.

[21] Liquibase Inc., «The Liquibase Community | The Database DevOps Community,» [En ligne]. Available: https://www.liquibase.org/.

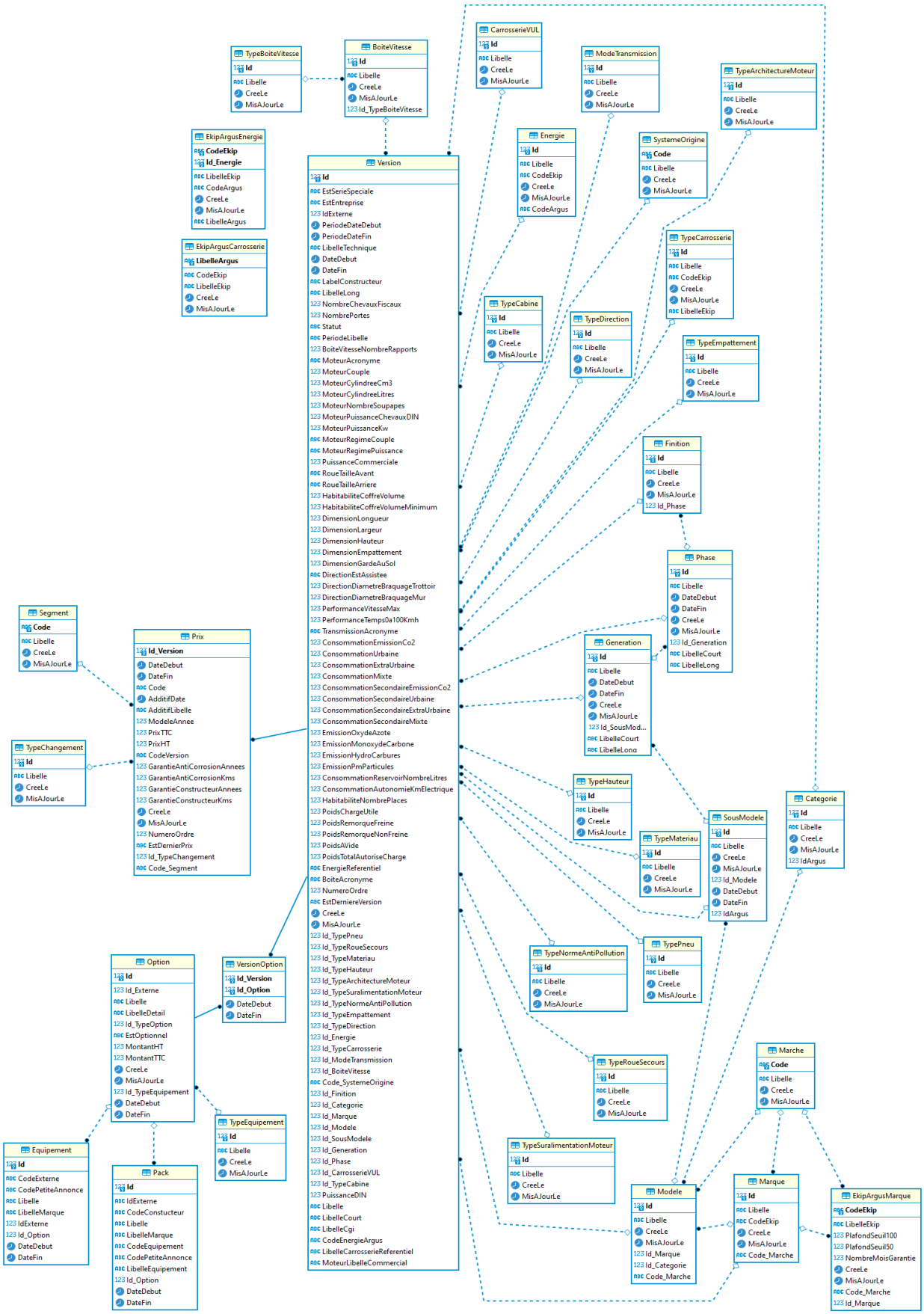
[22] Refactoring Guru, «Factory Method,» [En ligne]. Available: https://refactoring.guru/design-patterns/factory-method.

Table des Illustrations

Table des Annexes

ANNEXES

Annexe 3.2.3 : Diagramme du référentiel Argus enrichi par CGI Finance



Annexe 3.2.4.2 : Analyse quantitative des doublons d'id Argus dans le référentiel

```
with Comptages as (
    select v."IdExterne" , count(*) as nb_occurences
    from rfdp."Version" v
    group by v."IdExterne"
)
select
    (select count(distinct v."IdExterne") from rfdp."Version" v) as nb_id_argus,
    SUM(nb_occurences) as nb_lignes_total,
    avg(nb_occurences) as moyennes_occurences_id_argus,
    MIN(nb_occurences) as nb_minimal_id_argus,
    MAX(nb_occurences) as nb_maximal_id_argus,
    STDDEV(nb_occurences) as ecart_type_occurences_id_argus
from Comptages
```

Résultats 1

	nb_id_argus	nb_lignes_total	moyennes_occurences_id_argus	nb_minimal_id_argus	nb_maximal_id_argus	ecart_type_occurences_id_argus
1	124 799	318 320	2,5506614636	1	16	1,2255377116

Annexe 3.2.4.3 : Analyse qualitative des doublons d'id Argus dans le référentiel

```
select v."Id" , v."IdExterne" , v."PeriodeDateDebut" , v."PeriodeDateFin" , v."DateDebut" , v."DateFin" , v."LibelleLong"
from rfdp."Version" v
where v."IdExterne" = 1018485 or v."IdExterne" = 2374159
order by v."PeriodeDateDebut"
```

Version 1

	Id	IdExterne	PeriodeDateDebut	PeriodeDateFin	DateDebut	DateFin	LibelleLong
1	228 836	1 018 485	2009-01-01	2009-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
2	228 835	1 018 485	2010-01-01	2010-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
3	228 834	1 018 485	2011-01-01	2011-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
4	228 833	1 018 485	2012-01-01	2012-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
5	228 832	1 018 485	2013-01-01	2013-12-31	2009-05-01	2013-09-01	Génération II Phase Ph2 EEV 219 32S Tempmatik
6	356 981	2 374 159	2019-01-01	2019-12-31	2019-04-01	[NULL]	Génération III Phase Ph2 35S21 V9 Hi-Matic
7	356 982	2 374 159	2020-01-01	2020-12-31	2019-04-01	[NULL]	Génération III Phase Ph2 35S21 V9 Hi-Matic
8	356 983	2 374 159	2021-01-01	2021-12-31	2019-04-01	[NULL]	Génération III Phase Ph2 35S21 V9 Hi-Matic

Annexe 3.3.2 : Extrait des données véhicules du S.I.V.

id_technique	b5.tvv	b5.type_recep	b5.vin	b5.cnit	b5.masse_f1	b5.masse_f2	b5.masse_f3	b5.masse_f4	b5.poids_a_vide	b5.categorie	b5.genre	b5.carrosserie_ce	b5.carrosserie	b5.num_reception	b5.cylindree	b5.puissance_net_maxi	b5.energie	b5.puissance_admin
6044974898	DM2UFDA1R5NMFP7	CE	WF0RXXGCDRCL35794	M10FRDVP019W001	2060	2060	4060	1584	1509	M1	VP	AF	CI	e13*2001/116*0109*30	1997	103	GO	8
6045004198	SC9HR8/S	CE	VF7SC9HR8CA602888	M10CTRVP001Y335	1648	1648	2548	1165	1090	M1	VP	AB	CI	e2*2007/46*0003*20	1560	82	GO	5
6045073149	XP13M(a)KSP13F(MH)KSP130L-CHMRKW(1C)	CE	VNKKL3D330A065156	M10JT0VP004J434	1430	1430	2180	1090	1015	M1	VP	AF	CI	e11*2007/46*0152*02	998	51	ES	4
6045102378	S-DKN111A06D5B2BDA5	CE	W0L0SDL68D4047412	M10PELVP0355219	1625	1625	2455	1163	1088	M1	VP	AF	CI	e1*2001/116*0379*21	1229	63	ES	5
6045109840	CNJ105	CE	VF1CNJ10548226871	M10RENVP026M787	1360	1360	1710	939	864	M1	VP	AB	CI	PA-0031-19-01	1149	55	FE	4
6045119411	GDHB5P11M61BZ1	CE	TMAD351AAEJ091209	M10HMCVP000T930	1820	1820	3020	1381	1306	M1	VP	AB	CI	e11*2007/46*0337*03	1396	73.2	ES	6
6045137372	CR9S0H	CE	VF1CR9S0H48289226	M10RENVP308R934	1570	1570	2500	1055	980	M1	VP	AB	CI	PA-0022-19-01	1149	55	FE	5
6045155627	5RRL0H	CE	VF15RRL0H47490657	M10RENVP168X825	1658	1658	2558	1146	1071	M1	VP	AB	CI	e2*2001/116*0327*48	1461	66	GO	4
6045163000	JCF1MFTD7MD1A	CE	3C4PFACYXDT552103	M10FATVP008R662	2495	2495	3745	1940	1865	M1	VP	AF	CI	e11*2001/116*0145*08	1956	125	GO	10
6045163887	JKFJA6	CE	VF1JKFJA648115461	M10RENVP026F367	2605	2605	4105	1976	1901	M1	VP	AF	CI	e2*98/14*0265*41	1995	110	GO	8
6045175888	FLB0CS	CE	VF1FLB0CSDY478335	N10RENCT0173713	2940	2940	4940	1947	1872	N1	CTTE	BB	FOURGON	e2*2007/46*0014*15	1995	66	GO	7
6045180343	XP13M(a)KSP13F(MH)KSP130L-CHMRKW(1C)	CE	VNKKL3D370A056640	M10JT0VP004J434	1430	1430	2180	1090	1015	M1	VP	AF	CI	e11*2007/46*0152*02	998	51	ES	4
6045195390	4C5FS0/1	CE	VF34C5FS0DS048215	M10PGTVP0040914	1815	1815	3315	1332	1257	M1	VP	AF	CI	e2*2001/116*0362*35	1598	88	ES	7
6045209156	8D5FVA	CE	VF38D5FVACL086846	M10PGTVP0032299	2015	2015	3615	1485	1410	M1	VP	AA	CI	e2*2007/46*0080*09	1598	115	ES	9
6045210515	XA3(a)ALA30(WD)ALA30L-BWFGRW(3J)	CE	JTMBC31V20D113310	M10JT0VP004X096	2190	2190	4190	1693	1618	M1	VP	AF	CI	e6*2001/116*0105*08	2231	110	GO	9
6045212555	CA9HP0/S	CE	VF3CA9HP0CT153830	M10PGTVP004P999	1625	1625	2595	1142	1067	M1	VP	AB	CI	e2*2007/46*0070*07	1560	68	GO	5
6045219296	UKL-CSY115H0C0	CE	WMWSY110X0T514150	M10MN1VP000R774	1410	1410		1195	1120	M1	VP	AE	CABR	e1*2007/46*0369*07	1598	90	ES	7
6045226771	NC9HD8/1PS	CE	VF7NC9HD8DY504315	M10CTRVP004W948	1825	1825	3125	1280	1205	M1	VP	AB	CI	e2*2007/46*0040*12	1560	84	GO	6
6045234311	CNJ105	CE	VF1CNJ10548544448	M10RENVP026M787	1360	1360	1710	939	864	M1	VP	AB	CI	e2*2001/116*0359*32	1149	55	ES	4
6045238638	CCHMZ0	CE	VF3CCHMZ0DT019283	M10PGTVP0042431	1530	1530	2350	1050	975	M1	VP	AB	CI	e2*2007/46*0070*10	1199	60	ES	4
6045246349	6RABCGGBFM5FM52T088LLNVR07MG	CE	WVWZZZ6RZDY127668	M10VWGVP216Z559	1550	1550	2585	1070	995	M1	VP	AB	CI	e1*2001/116*0510*15	1390	63	ES	5
6045247712	WA8HR0/1	CE	VF3WA8HR0CE070430	M10PGTVP001D251	1642	1642	2242	1246	1171	M1	VP	AB	CI	e2*2001/116*0340*26	1398	50	GO	4
6045254333	KL1YYWBS1CC7	CE	KL1YF75Y9DK120889	M10CHEVP0011582	2295	2295	3795	1786	1711	M1	VP	AC	BREAK	e4*2007/46*0224*07	1998	120	GO	9
6045261342	BB2L05	CE	VF1BB2L0548273001	M10RENVP161L723	1460	1460	2250	1069	994	M1	VP	AB	CI	e2*98/14*0126*73	1149	55	ES	5
6045266137	HSD8T5	CE	UU1HSD8T548904454	M10DACVP000Z625	1750	1750	2850	1338	1263	M1	VP	AC	BREAK	e2*2001/116*0314*63	1598	75	EG	6
6045272204	CC8HR0/T	NA	VF3CC8HR0DT026448		1560	1560	1980	1186	1111	N1	CTTE		DERIV_VP	P-0017.12.00	1398	50	GO	6
6045283287	BSDBLK	CE	UU1BSDBLK48077805	M10DACVP000M454	1615	1615	2400	1237	1162	M1	VP	AB	CI	e2*2001/116*0314*61	1461	65	GO	5
6045284708	BZ1406	CE	VF1BZ140648223735	M10RENVP019P736	1813	1813	3113	1290	1215	M1	VP	AB	CI	e2*2001/116*0373*36	1461	81	GO	5
6045296267	8XSCBZAF1FM52T106R8X51LLS57MQEM2	CE	WAUZZZ8X6DB059810	M10AUDVP005P239	1555	1555	2585	1140	1065	M1	VP	AB	CI	e1*2007/46*0414*08	1197	63	ES	5
6045297941	JZ0903	CE	VF1JZ090348442484	M10RENVP0094786	1993	1993	3293	1505	1430	M1	VP	AB	CI	e2*2001/116*0379*27	1461	81	GO	6
6045303640	JZNA06	CE	VF1JZNA0648585891	M10RENVP321E908	2035	2035	3335	1505	1430	M1	VP	AB	CI	e2*2001/116*0379*27	1598	96	GO	7
6045306170	KL1MMHA121AA5	CE	KL1MF4819DC552266	M10CHEVP001N507	1367	1367		1044	969	M1	VP	AB	CI	e4*2007/46*0129*09	995	50	ES	4
6045318356	CNJ105	CE	VF1CNJ10548545265	M10RENVP026M787	1360	1360	1710	939	864	M1	VP	AB	CI	e2*2001/116*0359*32	1149	55	ES	4

b5.rapport_puiss_masse	b5.nb_places_assises	b5.nb_places_debout	b5.niv_sonore	b5.vitesse_moteur	b5.co2	b5.classe_env	b5.couleur	b4b.marque	b4b.denomination_comm	b4b.num_plaque	b4b.date_prem_immat	b4b.date_immat_siv	b4b.usages	b4b.mentions	b4b.date_ci
	5		75	2813	154	715/2007*459/2012EURO5	BLANC	FORD	KUGA	CM-977-GM	29/10/2012	29/10/2012			21/10/2022
	5		72	2700	99	715/2007*692/2008EURO5	GRIS FONCE	CITROEN	C3	CN-243-AZ	20/11/2012	20/11/2012			23/10/2022
	5		76	3750	111	715/2007*692/2008EURO5	ROUGE FONCE	TOYOTA	YARIS	CP-256-NW	07/01/2013	07/01/2013			21/10/2022
	5		84	3750	129	715/2007*566/2011EURO5	NOIR	OPEL	CORSA	CM-607-QB	07/11/2012	07/11/2012			21/10/2022
	4		85	3750	105	715/2007*566/2011EURO5	BLEU CLAIR	RENAULT	TWINGO	CQ-533-JN	31/01/2013	31/01/2013			21/10/2022
	5		79	3750	139	715/2007*566/2011EURO5	BLEU CLAIR	HYUNDAI	I 30	CQ-708-PL	07/02/2013	07/02/2013			22/10/2022
	5		78	4125	130	715/2007*566/2011EURO5	ROUGE CLAIR	RENAULT	CLIO	CP-203-BR	19/12/2012	19/12/2012			21/10/2022
	5		81	3000	90	715/2007*566/2011EURO5	GRIS FONCE	RENAULT	CLIO	CN-038-GH	27/11/2012	27/11/2012			25/10/2022
	7		78	3000	169	715/2007*692/2008EURO5	GRIS FONCE	FIAT	FREEMONT	CN-709-ZY	18/12/2012	18/12/2012			27/10/2022
	7		76	3000	150	715/2007*566/2011EURO5	BLEU CLAIR	RENAULT	ESPACE	CP-073-AK	18/12/2012	18/12/2012		PLMO (1,7)	21/10/2022
	3		76	2625	207	715/2007*566/2011EURO5	JAUNE CLAIR	RENAULT	TRAFIC	CM-251-ZY	19/11/2012	19/11/2012			21/10/2022
	5		76	3750	111	715/2007*692/2008EURO5	GRIS CLAIR	TOYOTA	YARIS	CN-400-CW	22/11/2012	22/11/2012			22/10/2022
	5		82	4500	147	715/2007*692/2008EURO5	BLANC	PEUGEOT	308	CS-066-EX	27/03/2013	27/03/2013			22/10/2022
	5		75	3750	164	715/2007*692/2008EURO5	MARRON FONCE	PEUGEOT	508	CN-092-TW	11/12/2012	11/12/2012			23/10/2022
	5		74	2700	159	715/2007*692/2008EURO5	MARRON FONCE	TOYOTA	RAV4	CN-942-VR	12/12/2012	12/12/2012			22/10/2022
	5		77	3000	98	715/2007*566/2011EURO5	ROUGE FONCE	PEUGEOT	208	CN-873-WE	12/12/2012	12/12/2012			27/10/2022
	2		78	3750	133	715/2007*566/2011EURO5	NOIR	MINI	MINI	CN-263-XZ	14/12/2012	14/12/2012			14/12/2012
	5		74	2700	104	715/2007*566/2011EURO5	GRIS FONCE	CITROEN	C4	CQ-700-JH	31/01/2013	31/01/2013			31/01/2013
	4		85	3750	105	715/2007*566/2011EURO5	NOIR	RENAULT	TWINGO	CR-961-EJ	26/02/2013	26/02/2013			26/02/2013
	5		76	3750	104	715/2007*566/2011EURO5	GRIS CLAIR	PEUGEOT	208	CR-850-GH	28/02/2013	28/02/2013			24/10/2022
	5		81	3750	139	715/2007*566/2011EURO5	NOIR	VOLKSWAGEN	POLO	CP-759-LM	17/12/2012	17/12/2012			03/01/2013
	5		77	3000	110	715/2007*692/2008EURO5	GRIS CLAIR	PEUGEOT	207	CQ-157-WJ	15/02/2013	15/02/2013			21/10/2022
	7		76	2850	139	715/2007*692/2008EURO5	BLANC	CHEVROLET	ORLANDO	CT-435-LM	03/05/2013	03/05/2013			24/10/2022
	5		82	4125	133	715/2007*566/2011EURO5	BLANC	RENAULT	CLIO	CQ-751-CY	24/01/2013	24/01/2013			09/08/2022
	5		78	3750	145	715/2007*630/2012EURO5	GRIS FONCE	DACIA	DUSTER	CR-981-XL	19/03/2013	19/03/2013			24/10/2022
0	2	0	77	3000	98		BLANC	PEUGEOT	208	CT-177-WT	17/05/2013	17/05/2013			24/10/2022
	5		77	2813	121	715/2007*566/2011EURO5	GRIS FONCE	DACIA	SANDERO	CQ-447-JF	31/01/2013	31/01/2013			31/01/2013
	5		76	3000	90	715/2007*566/2011EURO5	GRIS CLAIR	RENAULT	MEGANE	CQ-521-KA	01/02/2013	01/02/2013			26/10/2022
	5		78	3600	118	715/2007*566/2011EURO5	NOIR	AUDI	A1 SPORTBACK	CV-914-HR	31/05/2013	31/05/2013			21/10/2022
	5		77	3000	124	715/2007*566/2011EURO5	BEIGE CLAIR	RENAULT	MEGANE SCENIC	CR-862-YW	04/03/2013	21/03/2013			24/10/2022
	5		76	3000	114	715/2007*630/2012EURO5	INCONNU	RENAULT	MEGANE SCENIC	CV-798-MC	05/06/2013	05/06/2013			21/10/2022
	5		81	4800	119	715/2007*692/2008EURO5	NOIR	CHEVROLET	SPARK	CQ-064-TW	14/02/2013	14/02/2013			21/10/2022
	4		85	3750	105	715/2007*566/2011EURO5	GRIS CLAIR	RENAULT	TWINGO	CR-084-DG	25/02/2013	25/02/2013			22/10/2022
	3						INCONNU	CITROEN		CM-770-XE	15/11/2012	15/11/2012			24/10/2022

Annexe 3.4.2 : Fonction de tri des id argus dans la réconciliation Occazio

```
def execute(self) -> RapprochementIdArgusOut:
    """ Effectue le rapprochement entre le dictionnaire et une liste d'id argus.
    Gère les différents cas d'erreurs """
    try:
        ids_argus_rapproches = self.look_for_most_scoring_ids()
    except RapprochementException as e:
        rapprochement = RapprochementIdArgusOut(
            succes=False,
            ids_argus=[]
        )

        if isinstance(e, UnknownEnergyException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.energie_inconnue
        elif isinstance(e, UnknownBrandException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.marque_inconnue
        elif isinstance(e, UnknownModelException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.modele_inconnu
        elif isinstance(e, UnknownYearException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.annee_inconnue
        elif isinstance(e, VehicleTooOldException):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.vehicule_trop_vieux
        elif isinstance(e, NoResultWithFirstQueryException) \
                or isinstance(e, NoResultWhenFilteringByPositiveProbability) \
                or isinstance(e, NoResultWhenFilteringByPositiveProbability):
            rapprochement.type_succes = TypeSuccesRapprochementEnum.aucun_vehicule_matche

        return rapprochement

    rapp_id_argus = RapprochementIdArgusOut(
        succes=True,
        ids_argus=ids_argus_rapproches
    )

    if len(ids_argus_rapproches) == 1:
        rapp_id_argus.type_succes = TypeSuccesRapprochementEnum.certain
        return rapp_id_argus

    proba_id_argus_1 = ids_argus_rapproches[0].probabilite

    if proba_id_argus_1 >= .98:
        rapp_id_argus.type_succes = TypeSuccesRapprochementEnum.quasi_certain
        return rapp_id_argus

    proba_id_argus_2 = ids_argus_rapproches[1].probabilite
    if (proba_id_argus_1 - proba_id_argus_2) > self.seuil_diff_proba_1_2:  # par défaut 10%
        rapp_id_argus.type_succes = TypeSuccesRapprochementEnum.gap_id_argus1_et_idargus2
        return rapp_id_argus

    ecart_prix_neuf_id_argus_1_et_2 = abs(
        ids_argus_rapproches[0].prix_neuf_ttc - ids_argus_rapproches[1].prix_neuf_ttc
    ) / ids_argus_rapproches[1].prix_neuf_ttc

    if len(ids_argus_rapproches) <= 2:
        if ecart_prix_neuf_id_argus_1_et_2 <= self.seuil_ecart_rel_prxneuf:  # par défaut 5%
            rapp_id_argus.type_succes = TypeSuccesRapprochementEnum.petit_ecart_prix_neuf_top_2
            return rapp_id_argus
        else:
            rapp_id_argus.succes = False
            rapp_id_argus.type_succes = TypeSuccesRapprochementEnum.indecis
            return rapp_id_argus

    ecart_prix_neuf_id_argus_1_et_3 = abs(
        ids_argus_rapproches[0].prix_neuf_ttc - ids_argus_rapproches[2].prix_neuf_ttc
    ) / ids_argus_rapproches[2].prix_neuf_ttc

    if (ecart_prix_neuf_id_argus_1_et_2 <= self.seuil_ecart_rel_prxneuf) \
            & (ecart_prix_neuf_id_argus_1_et_3 <= self.seuil_ecart_rel_prxneuf):
        rapp_id_argus.type_succes = TypeSuccesRapprochementEnum.petit_ecart_prix_neuf_top_3
        return rapp_id_argus

    # proba_cumulees: par défaut 70%
    if (proba_id_argus_1 + proba_id_argus_2) > self.probas_cumulees \
            and ecart_prix_neuf_id_argus_1_et_2 <= self.seuil_ecart_rel_prxneuf:
        rapp_id_argus.type_succes = TypeSuccesRapprochementEnum.petit_ecart_prix_neuf_top_2
        return rapp_id_argus

    rapp_id_argus.succes = False
    rapp_id_argus.type_succes = TypeSuccesRapprochementEnum.indecis
    return rapp_id_argus
```

Annexe 4.1 : Tableau de corrélation terminologique entre le métier et le technique

Vue Métier	Vue Technique	Commentaires
Gamme Finition Version	Id Argus	Le but de l'algorithme de réconciliation est de trouver en résultat ce niveau de granularité au niveau du véhicule.
	Finition	Colonne « finition » de la base référentiel Argus.
Caractéristique de véhicule	Dimension	La dimension est une subdivision d'un véhicule avec un type, une valeur et éventuellement une distance représentant une caractéristique de véhicule.
Référentiel Argus	Catalogue Constructeur	D'un point de vue technique, le référentiel Argus est appelé Catalogue Constructeur

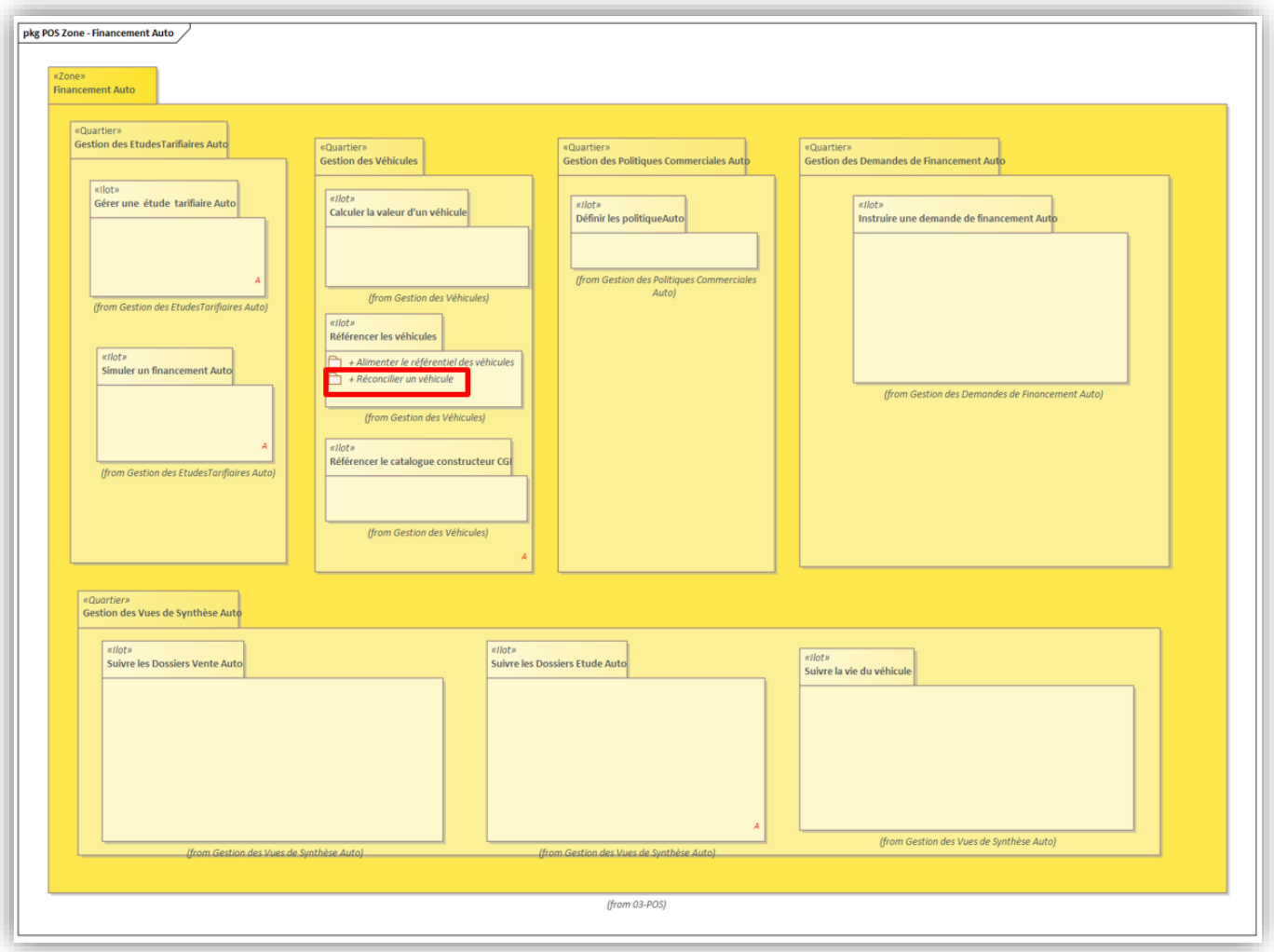
Annexe 4.2.1.a : Extrait de données de ReezoCar

partTC	reference	modele	codeMarque	dateProduction	dateMiseEnCirculation	kilometrage	puissanceFiscale	co2	Portes	Finition	Automatique	Gamme	immatriculation	Version	nb_rapports	codeEnergie	codeConsommation	makeApplicable	enStock	certificatScellPartenaire	Puissance_din
13499	RZCATSELDUZDBDCB5D1ED	CORSA	OPE		01/06/2020	16092	4.0	122 g/km	5.0	CORSA-E EDITION	FAUX	CORSA	REUDZDB0ZC	1.2 75PS/55KW EDITION	5 rapports	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
53323	RZCATSELDTA4AFC96A2B	TT	AUD		01/06/2023	10	14.0	182 g/km	3.0	NOT_FOUND	VRAI	TT	REUTDA4AFC	2.0 45 TFSI S TRONIC S LINE COUPE	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
14999	RZCATSELEC48A7CFEB1	2008	PEU		01/11/2013	94416	nan	nan/km	5.0	CROSSWAY	FAUX	2008	REUEC48FA7	1.2 PURETECH 110 S&S CROSSWAY	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
7998	RZCATSELE5F7A25D2A8CC	CLIO	REN		01/02/2018	98492	4.0	85 g/km	5.0	NOT_FOUND	FAUX	CLIO	REUE5F7A25	1.5 DCI 75 LIMITED - 18	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
26049	RZCATSELDZ0DF67A7E2A51	X1	BMW		01/09/2019	97870	0.0	114 g/km	5.0	XLINE	VRAI	X1	REUZ0DF6A7	2.0 SDRIVE18D XLINE AUTO	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
32499	RZCATSELDB4721ED7D644	3008	PEU		01/09/2020	50417	10.0	30 g/km	5.0	ALLURE	VRAI	3008	REUB4721ED	nan	nan	nan	T0D6A9Z026	VRAI	VRAI	VRAI	
23498	RZCATSELD6A91D95CA70E	208	PEU		01/08/2021	29239	nan	nan/km	5.0	ACTIVE BUSINESS	VRAI	208	REUD6A91D9	ELECTRIQUE ACTIVE BUSINESS	1 rapports	ELE	T0D6A9Z026	VRAI	VRAI	VRAI	
27258	RZCATSELD37FDD31E6E4C	TALISMAN	REN		01/05/2021	32887	nan	nan/km	4.0	INTENS	VRAI	TALISMAN	REUD73F9D03	1.3 TCE 160 FAP INTENS EDC	7 rapports	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
53736	RZCATSELDBF4D9B0B4967	TT	AUD		01/05/2023	10	14.0	182 g/km	3.0	NOT_FOUND	VRAI	TT	REUBF4D9B0	2.0 45 TFSI S TRONIC S LINE COUPE	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
16999	RZCATSELD9CFCA7B0FCBF	GRAND SCENIC	REN		01/05/2017	120533	nan	30 g/km	5.0	INTENS	VRAI	GRAND SCENIC	REUD9CFCA7	1.6 DCI 130 INTENS EDC	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
81599	RZCATSELDB0D5D5E5582	I4	BMW		01/12/2022	9933	47.0	nan/km	5.0	M50	VRAI	I4	REUB0D0D5E	BEV 84KWH M50I GRAN COUPE	1 rapports	ELE	T0D6A9Z026	VRAI	VRAI	VRAI	
23498	RZCATSELD6D7D8E64EF0C	KANGOO	REN		01/05/2022	2737	nan	nan/km	5.0	KANGOO	FAUX	KANGOO	REUD6D7D8E	1.5 DCI 95 BLUE ZEN	6 rapports	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
25960	RZCATSELDB6FD1253573	SERIE 1	BMW		01/06/2021	41281	nan	nan/km	5.0	EDITION SPORT	VRAI	SERIE 1	REUDB6FD12	nan	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
32998	RZCATSELDA1BAA7D5F380	CLASSE CL	MER		01/11/2019	59500	nan	nan/km	nan	NOT_FOUND	VRAI	CLASSE CL	REUDA1BAA7	nan	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
31249	RZCATSELDA2B0392CC12	ARKANA	REN		01/06/2022	14945	nan	nan/km	5.0	INTENS	VRAI	ARKANA	REUDA2B039	nan	nan	HYS	T0D6A9Z026	VRAI	VRAI	VRAI	
16999	RZCATSELDF6E4F013E077A	CLIO	REN		01/06/2020	15966	nan	nan/km	5.0	BUSINESS	FAUX	CLIO	REUDF6E4F0	1.5 DCI BLUE BUSINESS	6 rapports	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
52000	RZCATSELDB8DDC0E83AD	CLASSE GT	MER		01/06/2022	50	74.0	nan/km	nan	NOT_FOUND	VRAI	CLASSE GT	REUB8DDC0E	nan	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
37998	RZCATSELDBC150A0B81B48	5008	PEU		01/07/2020	41029	10.0	125 g/km	5.0	GT	VRAI	5008	REUBC150A0	2.0 BLUEHDI 180 S&S AUTO GT	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
15965	RZCATSELDB0FF93E0EE1	C3	CIT		01/03/2021	17483	4.0	121 g/km	5.0	FEEL BUSINESS	FAUX	C3	REUB0FF93E	nan	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
27900	RZCATSELDF7386EF0591	TIGUAN	VOK		01/02/2020	48160	8.0	122 g/km	5.0	CONFORTLINE	FAUX	TIGUAN	REUDF7386E	2.0 TDI 150 CONFORTLINE	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
33999	RZCATSELDF5FDD5B27D83D	CLASSE A	MER		01/11/2020	13239	8.0	125 g/km	nan	AMG LINE	FAUX	CLASSE A	REUDF5FDD5	nan	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
15959	RZCATSELDF967B83E4D075	CLIO	REN		01/05/2021	31144	5.0	108 g/km	5.0	BUSINESS	FAUX	CLIO	REUDF967B3	1.0 TCE 100 GPF BUSINESS	nan	GPL	T0D6A9Z026	VRAI	VRAI	VRAI	
39999	RZCATSELDF02F0181B1A	MASTER	REN		01/11/2020	20793	nan	nan/km	nan	NOT_FOUND	FAUX	MASTER	REUDF02F03	nan	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
61999	RZCATSELDF8C24B8E5A46B	Q8	AUD		01/06/2019	142000	18.0	157 g/km	nan	AVUS EXTENDED	VRAI	Q8	REUDF8C24B	nan	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
36240	RZCATSELDF9FF13D0F54DF	500X	FIA		01/08/2022	10	nan	nan/km	5.0	NOT_FOUND	VRAI	500X	REUDF9FF13	1.5 MHEV FIRJ 130 DCT RED DOLCEVITA	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
42999	RZCATSELDF9D0D45CC7B0D	208	AUD		01/09/2022	10	8.0	132 g/km	5.0	S LINE	FAUX	Q2	REUDF9D0D4	nan	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
20999	RZCATSELDF9D0D45CC7B0D	208	PEU		01/06/2022	7000	nan	nan/km	5.0	ALLURE	FAUX	208	REUDF9D0D4	nan	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
25453	RZCATSELDF9AAE5A4CD0F	SPORTAGE	KIA		01/07/2023	10	0.0	148 g/km	nan	NOT_FOUND	FAUX	SPORTAGE	REUDF9AAE5	nan	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
22199	RZCATSELDF9D9CCE5928	CORSA	OPE		01/08/2022	150	5.0	106 g/km	5.0	ELEGANCE BUSINESS	FAUX	CORSA	REUDF9D9CC	1.5 DIESEL 100PS ELEGANCE BUSINESS	6 rapports	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
30999	RZCATSELDF9B62D8B65B8	DS	DS		01/01/2022	33701	7.0	127 g/km	nan	CHIC	VRAI	DS3	REUDF9B62D	nan	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
26999	RZCATSELDF9E4DE1DCC888	GRAND SCENIC	REN		01/06/2021	29427	nan	nan/km	nan	BUSINESS	FAUX	GRAND SCENIC	REUDF9E4DE	nan	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
6999	RZCATSELDF9B67AC5B67A9	CLIO	REN		01/06/2016	81797	nan	nan/km	5.0	NOT_FOUND	FAUX	CLIO	REUDF9B67A	1.5 DCI 90 ENERGY TREND	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
33999	RZCATSELDF9B7A2782E3C	GRANDLAND	OPE		01/08/2022	6000	7.0	144 g/km	5.0	NOT_FOUND	VRAI	GRANDLAND	REUDF9B7A2	1.2 ECOTEC TURBO 96KW GS LINE AUTO	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
14999	RZCATSELDF9D4B4A99E91	208	PEU		01/08/2019	72672	nan	nan/km	5.0	SIGNATURE	FAUX	208	REUDF9D4B4	1.5 BLUEHDI 100 S&S SIGNATURE	5 rapports	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
12999	RZCATSELDF9D67795953	MITO	ALF		01/10/2017	75775	nan	nan/km	3.0	IMOLA	FAUX	MITO	REUDF9D677	1.3 JTDM 95 S/S IMOLA	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
19799	RZCATSELDB55B8DC1E629	CLIO	REN		01/12/2021	20750	nan	nan/km	nan	NOT_FOUND	VRAI	CLIO	REUDB55B8D	nan	nan	HYS	T0D6A9Z026	VRAI	VRAI	VRAI	
49964	RZCATSELDA5B95A9E0C0	Q5	AUD		01/07/2022	22393	11.0	148 g/km	5.0	ADVANCED	VRAI	Q5	REUDA5B95A	2.0 40 TDI 204 QTT S TRONIC 7 DESIGN	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
9290	RZCATSELD06C3861EB54	ION	PEU		01/08/2018	54376	4.0	nan/km	5.0	NOT_FOUND	VRAI	ION	REUD06C3861	ELECTRIQUE	nan	ELE	T0D6A9Z026	VRAI	VRAI	VRAI	
35068	RZCATSELD0528AD2AD44	2008	PEU		01/05/2023	10	7.0	135 g/km	nan	GT	VRAI	2008	REUD0528AD	nan	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
43288	RZCATSELD0560563E80A4	Q3	AUD		01/02/2021	28876	11.0	135 g/km	5.0	NOT_FOUND	VRAI	Q3	REUD0560563	2.0 40 TDI 200 S TRONIC QUATTRO DESIGN	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
34790	RZCATSELD0C0D83B09CFC	X3	BMW		01/12/2019	72171	6.0	131 g/km	5.0	LOUNGE	VRAI	X3	REUD0C0D83	2.0 SDRIVE18D LOUNGE AUTO	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
18900	RZCATSELD0F9C7B0F5F	JUKE	NIS		01/02/2021	30719	nan	nan/km	5.0	N-CONNECTA	FAUX	JUKE	REUD0F9C27	1.0 DIG-T 114 N-CONNECTA	6 rapports	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	
84900	RZCATSELD7252BB83389F	X5	BMW		01/01/2021	45995	nan	nan/km	nan	M SPORT	VRAI	X5	REUD7252BB	nan	nan	HYS	T0D6A9Z026	VRAI	VRAI	VRAI	
33999	RZCATSELD3790985501D2	PRIMASTAR	NIS		01/07/2022	70	nan	nan/km	nan	N-CONNECTA	FAUX	PRIMASTAR	REUD3790098	nan	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
28999	RZCATSELDCEB7B1B54A98	E-NIRO	KIA		01/10/2020	36845	11.0	nan/km	nan	PREMIUM BUSINESS	VRAI	E-NIRO	REUDCEB7B1	nan	nan	ELE	T0D6A9Z026	VRAI	VRAI	VRAI	
92999	RZCATSELDZ7D025D44BE8	XC90	VOL		01/03/2023	6000	20.0	31 g/km	nan	ULTIMATE CHROME	VRAI	XC90	REUZ7D025D	nan	nan	nan	T0D6A9Z026	VRAI	VRAI	VRAI	
18949	RZCATSELD7169B2C9324	PARTNER	PEU		01/11/2019	49472	nan	nan/km	nan	NOT_FOUND	FAUX	PARTNER	REUD7169B2	nan	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
56399	RZCATSELD31B0B0A4E0FC	X4	BMW		01/09/2019	80937	21.0	170 g/km	5.0	NOT_FOUND	VRAI	X4	REUD31B0B0	3.0 M40D AUTO	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
41990	RZCATSELD88B4CF5B8C5D	CLASSE CL	MER		01/05/2023	8000	nan	nan/km	nan	NOT_FOUND	VRAI	CLASSE CL	REUD88B4CF	nan	nan	HYS	T0D6A9Z026	VRAI	VRAI	VRAI	
51390	RZCATSELDB0CF05D2926	X4	BMW		01/10/2019	31706	10.0	131 g/km	5.0	XLINE	VRAI	X4	REUDB0CF05	2.0 XDRIVE20D XLINE AUTO	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
21990	RZCATSELDE6549A87A2DA8	A4	AUD		01/11/2017	69000	8.0	104 g/km	4.0	NOT_FOUND	VRAI	A4	REUDE6549A	2.0 TDI ULTRA 150	nan	GSL	T0D6A9Z026	VRAI	VRAI	VRAI	
23182	RZCATSELDD7D9445018D	CLASSE G	MER		01/09/2021	28900	51.0	nan/km	nan	AMG	VRAI	CLASSE G	REUDD7D94A	nan	nan	ESS	T0D6A9Z026	VRAI	VRAI	VRAI	

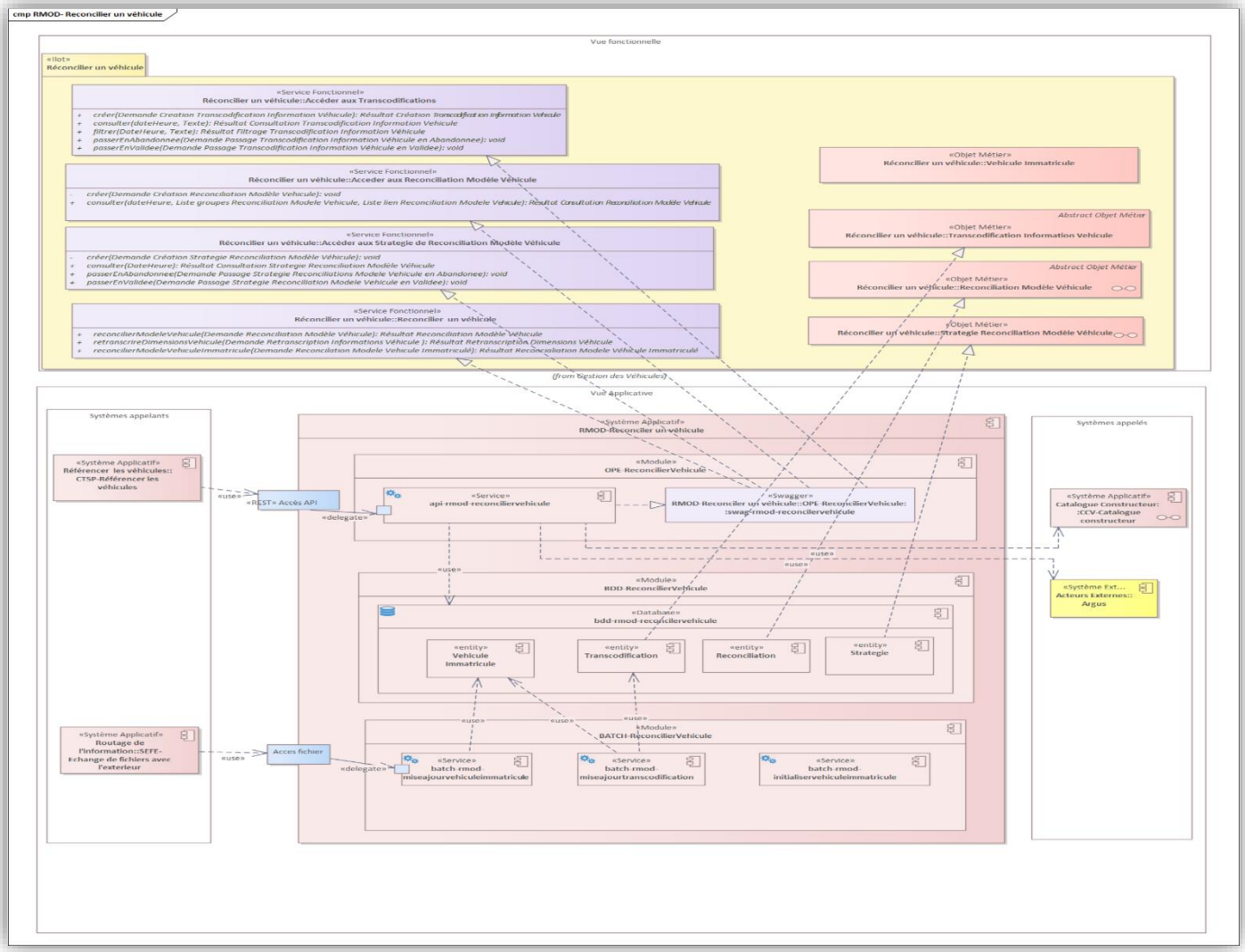
Annexe 4.2.1.b : Extrait de données de Planète VO

no_vehicule	source	groupe	id_argus_attendu	marque	modèle	date_mec	energie	boite_vitesse	type_boite_vitesse	version	puissance_fiscale	puissance_din	nb_places	nb_portes	emission_co2	cylindree	poids	carosserie
1	P/O		2036304	LAND-ROVER	Evoque	Diesel	Essence		Automatique	Evoque 2.0 D 150ch R-Dynamic SE AWD BVA	8	150	5	5	143	1999	Break	
2	P/O		1113909	SEAT	Ibiza	jeudi 10 mars 2011	Essence		Manuelle	Ibiza 1.2 TSI 105ch Style 5cv 5p	5	105	5	5	119	1197	Berline	
3	P/O		2009557	AUDI	A1 Sportback	lundi 12 avril 2019	Essence		Manuelle	A1 Sportback 30 TFSI 116ch S line	6	116	5	5	111	999	Berline	
4	P/O		2677320	VOLVO	C40	jeudi 6 avril 2023	Electrique		Automatique	C40 Recharge 231ch Plus	6	231	5	0	0		S.U.V.	
5	P/O		2336481	PEUGEOT	208	vendredi 31 mars 2023	Essence		Automatique	208 1.2 PureTech 130ch S&S GT E6.3	7	130	5	5	122	0	Berline	
6	P/O		2643976	BMW	Série 3 Touring	vendredi 31 mars 2023	Hybride rechargeable : Essence/Electrique		Automatique	Série 3 Touring 330e xDrive 292ch M Sport	10	184	5	5	35	0	Break	
7	P/O		2637500	BMW	Série 2 Active Tourer	lundi 3 avril 2023	Hybride rechargeable : Essence/Electrique		Manuelle	Série 2 Active Tourer 225e 245ch xDrive M Sport DKG7	7	136	5	14	0		Monospace	
8	P/O		2668837	FIAT	500	vendredi 31 mars 2023	Electrique		Automatique	500 e 118ch (RED)	3	118	4	0	0		Berline	
9	P/O		2670824	BMW	Série 1	vendredi 31 mars 2023	Diesel		Manuelle	Série 1 116dA 116ch M Sport DKG7	6	116	5	5	121	0	Berline	
10	P/O		2523343	PEUGEOT	208	vendredi 31 mars 2023	Diesel		Manuelle	208 1.5 BlueHDi 100ch S&S Active Pack	5	102	5	5	107	0	Berline	
11	P/O		2693600	TOYOTA	Aygo X	jeudi 6 avril 2023	Essence		Manuelle	Aygo X 1.0 VVT-i 72ch Design S-CVT MY23	4	72	4	5	112	0	Berline	
12	P/O		2310427	HYUNDAI	i30 Fastback	vendredi 31 mars 2023	Essence/Micro-Hybride		Manuelle	i30 Fastback 1.5 T-GDi 160ch Hybrid 48V N Line DCT-7	8	160	5	5	138	0	Berline	
13	P/O		2650742	FIAT	500C	vendredi 28 avril 2023	Essence/Micro-Hybride		Manuelle	500C 1.0 70ch BSG S&S Dolcevita Special Edition	4	69	4	2	108	0	Cabriolet	
14	P/O		2294149	KIA	Rio	vendredi 28 avril 2023	Essence/Micro-Hybride		Manuelle	Rio 1.0 T-GDI 120ch MHEV GT-Line Premium DCT7	6	120	5	5	125	0	Berline	
15	P/O		2640980	SEAT	Ateca	vendredi 28 avril 2023	Diesel		Manuelle	Ateca 2.0 TDI 150ch Start&Stop Xperience DSG7	8	150	5	5	129	0	S.U.V.	
16	P/O		2691617	FIAT	500e	vendredi 28 avril 2023	Electrique		Automatique	500e 118ch Pack Confort & Style	3	118	4	2	0	0	Berline	
17	P/O		2578264	FIAT	500	vendredi 28 avril 2023	Electrique		Automatique	500 e 118ch hôtre	3	118	4	3	0	0	Berline	
18	P/O		2675743	KIA	XCeed	vendredi 28 avril 2023	Essence		Manuelle	XCeed 1.5 T-GDI 160ch GT-Line Premium	8	160	5	5	141	0	Break	
19	P/O		2391161	HYUNDAI	Tucson	vendredi 28 avril 2023	Hybride rechargeable : Essence/Electrique		Automatique	Tucson 1.6 T-GDi 265ch PHEV Creative BVA6 HTRAC	10	180	5	5	31	0	S.U.V.	
20	P/O		2578264	FIAT	500	vendredi 28 avril 2023	Electrique		Automatique	500 e 118ch hôtre	3	118	4	3	0	0	Berline	
21	P/O		2650742	FIAT	500C	samedi 29 avril 2023	Essence/Micro-Hybride		Manuelle	500C 1.0 70ch BSG S&S Dolcevita Special Edition	4	69	4	2	108	0	Cabriolet	
22	P/O		2550994	KIA	Ceed	vendredi 28 avril 2023	Essence		Manuelle	Ceed 1.5 T-GDI 160ch GT Line Premium DCT7	8	160	5	5	133	0	Berline	
23	P/O		2287390	CITROEN	Grand C4 SpaceTourer	vendredi 10 juin 2022	Diesel		Manuelle	Grand C4 SpaceTourer BlueHDi 130ch S&S Feel E6.d	7	131	5	7	128	0	Monospace	
24	P/O		2659831	FIAT	Scudo Fg	vendredi 28 avril 2023	Diesel		Manuelle	Scudo Fg M 1.5 BlueHDi 120ch S&S Pro Lounge Connect	5	120	3	4	178	0	Fourgon	
25	P/O		2310930	VOLKSWAGEN	T-Cross	vendredi 28 avril 2023	Essence		Manuelle	T-Cross 1.0 TSI 110ch R-Line	6	110	5	5	134	0	S.U.V.	
26	P/O		2591130	SUZUKI	Vitara	vendredi 3 juin 2022	Hybride : Essence/Electrique		Automatique	Vitara 1.5 Dualjet Hybrid 102ch Style Auto	5	115	5	5	121	1462	Break	
27	P/O		1558032	FIAT	Fiorino	vendredi 7 septembre 2	Essence		Manuelle	Fiorino 1.4 i.e. 77ch Pack Clim +	8	78	2	3	161	1368	Fourgonnette	
28	P/O		2293907	KIA	Picanto	dimanche 11 juin 2023	Essence		Manuelle	Picanto 1.0 DPi 67ch Active	4	67	5	5	115	0	Berline	
29	P/O		2542197	VOLKSWAGEN	Taigo	samedi 20 août 2022	Essence		Manuelle	Taigo 1.5 TSI 150ch R-Line DSG7	8	150	5	5	138	1498	Break	
30	P/O		2247145	LAND-ROVER	Defender	mercredi 1 juillet 2020	Essence		Automatique	Defender 110 3.0 P400 First Edition	29	400	5	5	220	2996	Break	
31	P/O		2196209	CITROEN	C4 Cactus	vendredi 17 janvier 202	Diesel		Manuelle	C4 Cactus BlueHDi 100ch S&S Feel E6.d	5	102	5	5	92	1499	Berline	
32	P/O		2004963	SUZUKI	Swift	lundi 13 juillet 2020	Essence		Manuelle	Swift 1.2 Dualjet Hybrid 90ch Pack Eurodot-T	4	90	5	5	94	1242	Berline	
33	P/O		2324393	MERCEDES-BENZ	Classe E Cabriolet	vendredi 18 février 202	Diesel		Manuelle	Classe E Cabriolet 400 d 339ch AMG Line 4Matic 9G-Tron	21	330	4	2	178	2925	Cabriolet	
34	P/O		2573108	FIAT	Doblo Cargo	lundi 16 mai 2022	Diesel		Manuelle	Doblo Cargo 1.6 Multijet 105ch 705kg Business	5	105	3	3	152	0	Fourgonnette	
35	P/O		2271647	SEAT	Leon	jeudi 29 avril 2021	Essence		Manuelle	Leon 1.0 TSI 110ch Urban	6	110	5	5	123	999	Berline	
36	P/O		2353038	SEAT	Ateca	vendredi 17 février 202	Essence		Manuelle	Ateca 1.5 TSI 150ch Start&Stop Style DSG	8	150	5	5	149	1498	Break	
37	P/O		2117158	HYUNDAI	Ioniq	jeudi 8 avril 2021	Hybride		Manuelle	Ioniq Hybrid 141ch Executive	5	105	5	5	102	1580	Berline	
38	P/O		2333625	CUPRA	Formentor	mercredi 21 avril 2021	Essence		Manuelle	Formentor 2.0 TSI 310ch VZ DSG7 4Drive	20	310	5	5	192	1984	S.U.V.	
39	P/O		2271842	SEAT	Leon	lundi 14 juin 2021	Essence		Manuelle	Leon 1.5 eTSI 150ch Xcellence DSG7	8	150	5	5	131	1498	Berline	
40	P/O		2311151	OPEL	Combo Life	vendredi 29 avril 2022	Diesel		Manuelle	Combo Life L1H1 1.5 D 100ch BVA6	5	100	5	5	135	0	Break	
41	P/O		2391162	HYUNDAI	Tucson	mercredi 7 septembre 2	Hybride		Automatique	Tucson 1.6 T-GDi 265ch PHEV HTRAC Executive BVA6	10	180	5	5	0	1598	Break	
42	P/O		2282149	HYUNDAI	i20	mercredi 10 août 2022	Essence/Micro-Hybride		Manuelle	i20 1.0 T-GDi 100ch Hybrid Creative DCT-7	5	100	5	5	116	998	Berline	
43	P/O		2316706	ISUZU	D-Max	vendredi 11 février 202	Diesel		Automatique	D-Max 1.9 Ddi 164ch N60 8 Speed A/T	6	163	4	2	0	1898	4x4 Tout terrain	
44	P/O		2391746	BMW	Série 4 Cabriolet	lundi 30 janvier 2023	Essence		Automatique	Série 4 Cabriolet 420A 184ch M Sport	10	184	4	2	152	0	Cabriolet	
45	P/O		1985662	PORSCHE	Macan	mardi 22 janvier 2019	Essence		Manuelle	Macan 2.0 245ch PDK	15	245	5	5	185	0	Break	
46	P/O		1226359	PORSCHE	Macan	mardi 9 août 2016	Essence		Manuelle	Macan 3.6 V6 400ch Turbo PDK	29	400	5	5	208	3604	Break	
47	P/O		2136174	PORSCHE	Cayenne Coupe	vendredi 4 septembre 2	Hybride rechargeable : Essence/Electrique		Manuelle	Cayenne Coupe 3.0 V6 462ch E-Hybrid Eurod6d-T-EVAP-IS	22	340	4	5	73	2995	Break	
48	P/O		968735	CITROEN	C4		Diesel		Manuelle	C4 1.6 HDi 110 FAP Exclusive BMP6	6	110	5	5	120	1560	Berline	
49	P/O		881847	TOYOTA	Doblo	jeudi 20 avril 2006	Diesel		Manuelle	Doblo 1.9 Multijet 8v 105ch Emotion	6	105	5	5	153	1910	Break	
50	P/O		938753	TOYOTA	Aygo		Essence		Manuelle	Aygo 1.0 VVTi 68ch Confort 5p	4	68	5	5	108	998	Berline	
51	P/O		956722	KIA	Rio	lundi 14 janvier 2008	Diesel		Manuelle	Rio 1.5 CRDi 110 Active	6	110	5	5	119	1493	Berline	

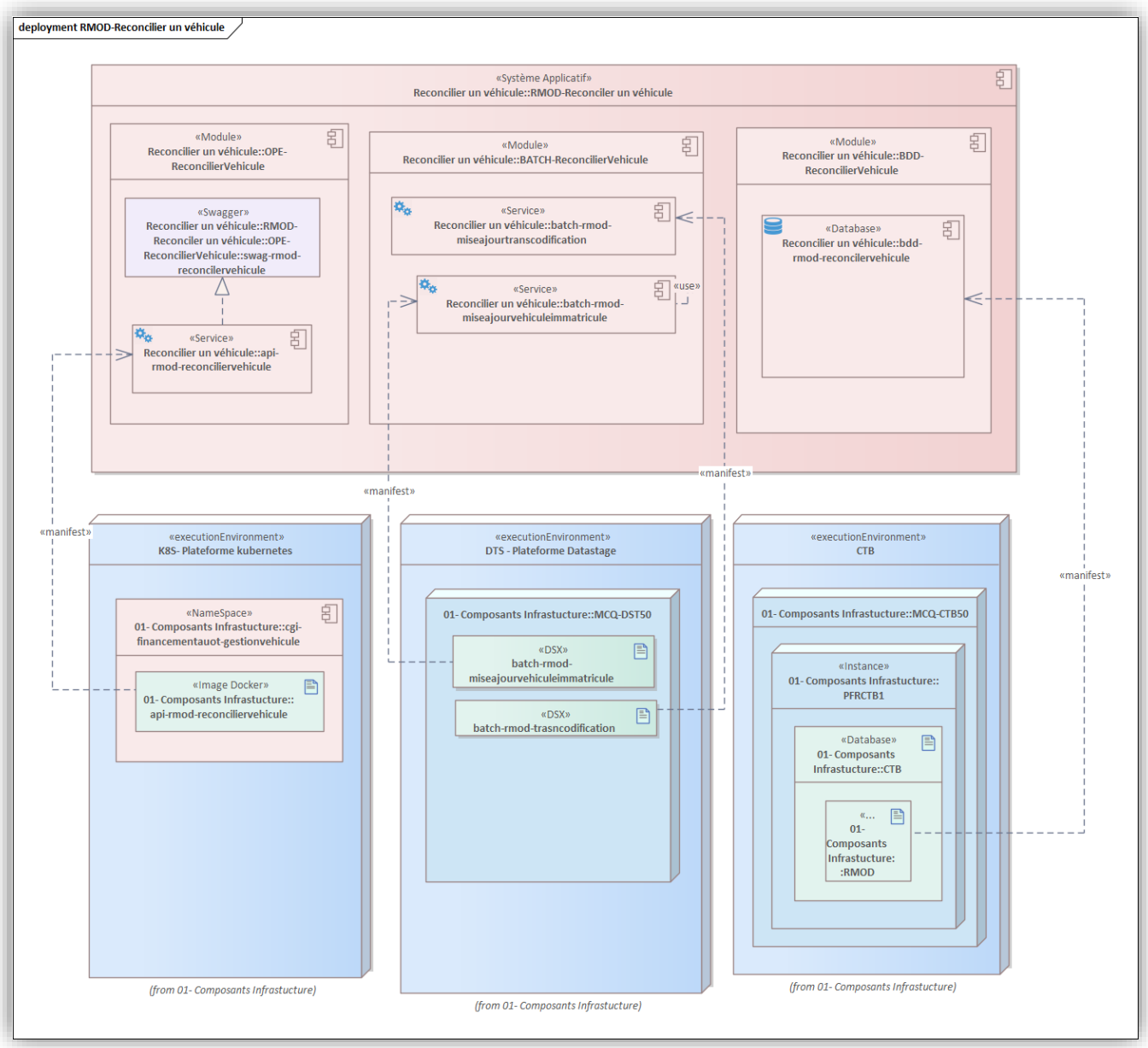
Annexe 4.3.1 : Périmètre de l'application RMOD au sein du SI de CGI Finance



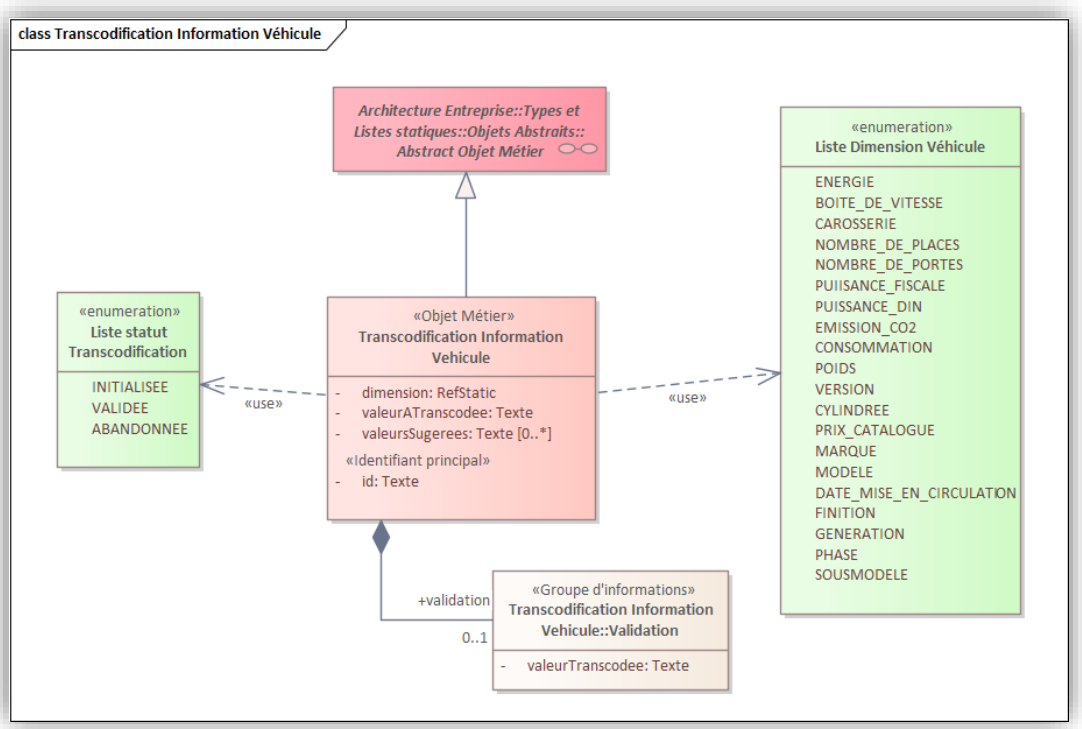
Annexe 4.3.2 : Schéma de liaison entre architecture fonctionnelle et applicative



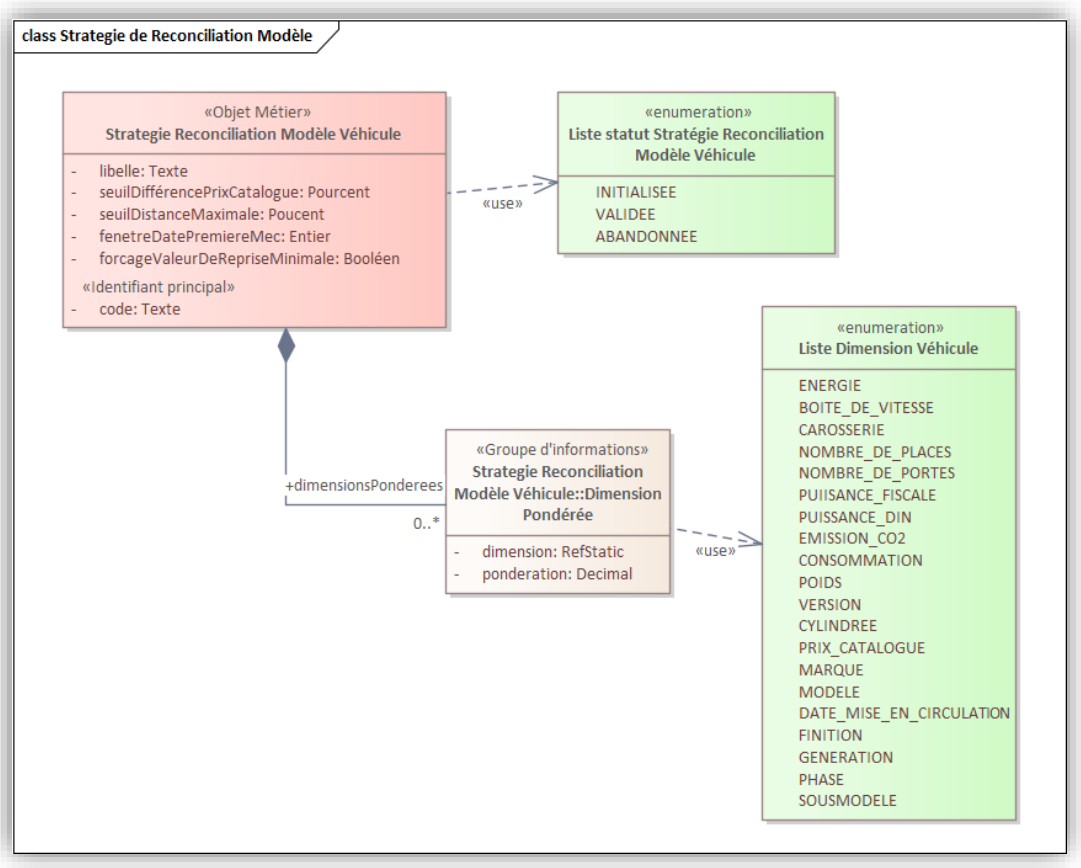
Annexe 4.3.5 : Schéma de liaison entre architecture applicative et technique



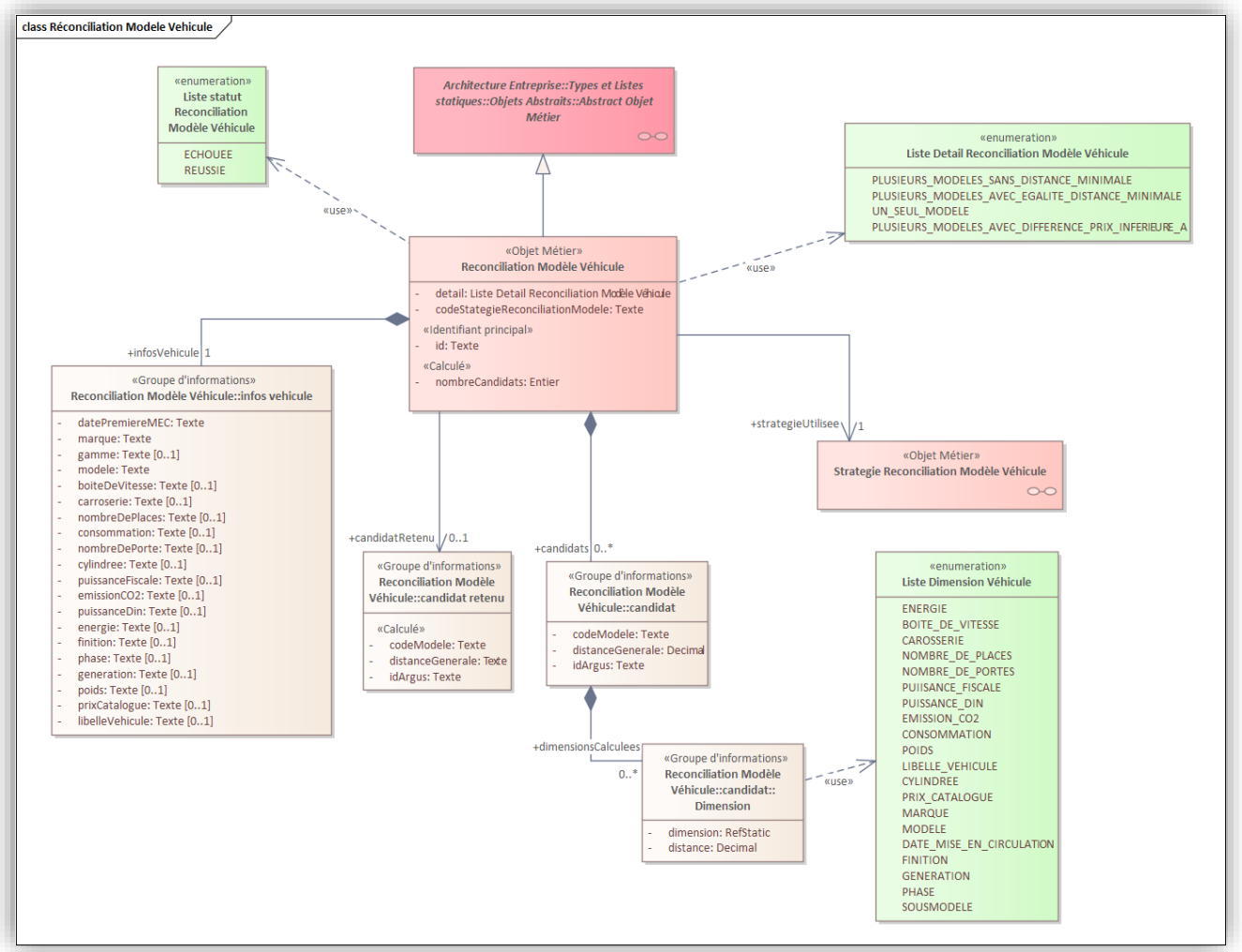
Annexe 4.4.2.a : Objet Métier Transcodification Information Véhicule



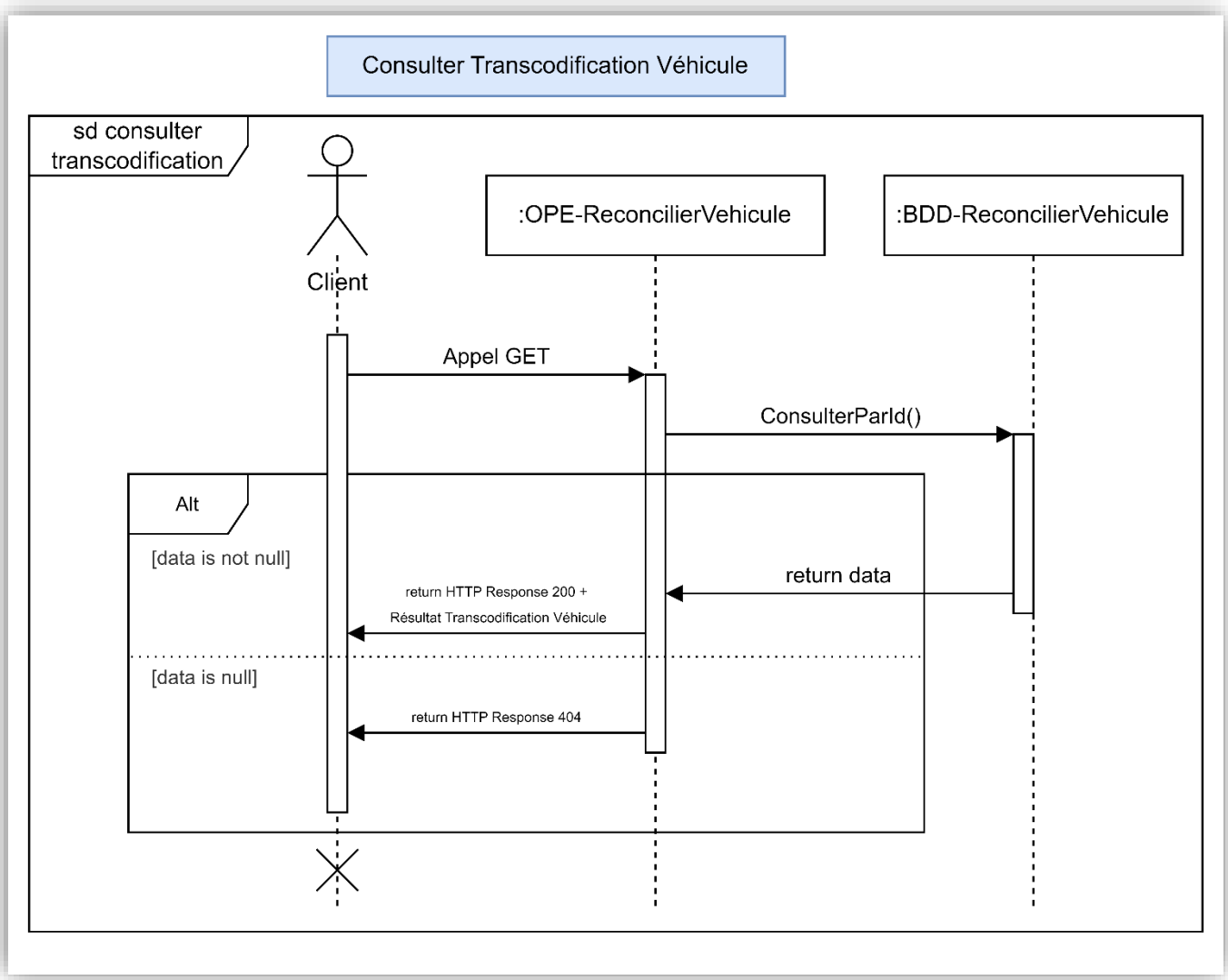
Annexe 4.4.2.b : Objet Métier Stratégie Réconciliation Modèle Véhicule



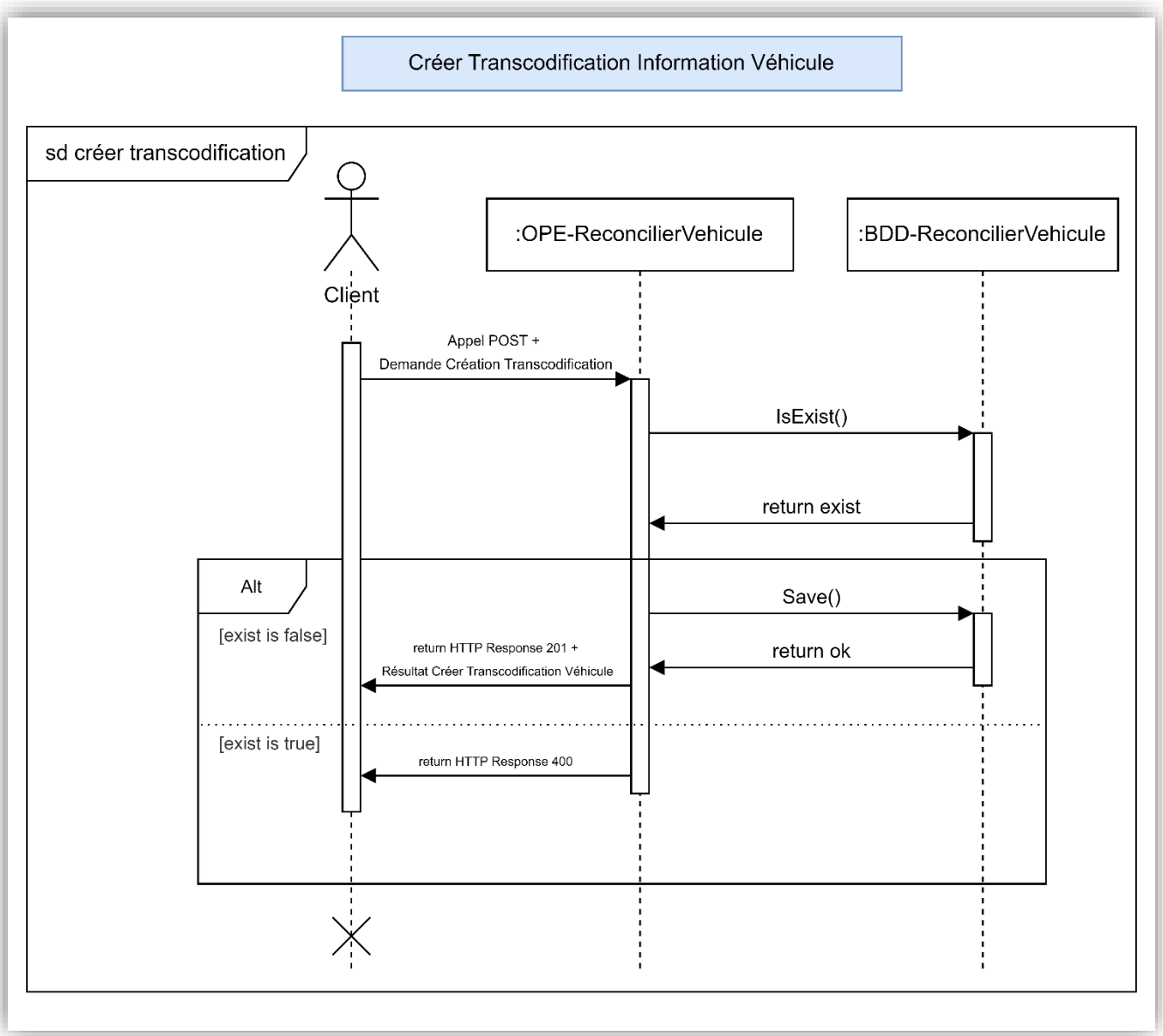
Annexe 4.4.2.c : Objet Métier Réconciliation Modèle Véhicule



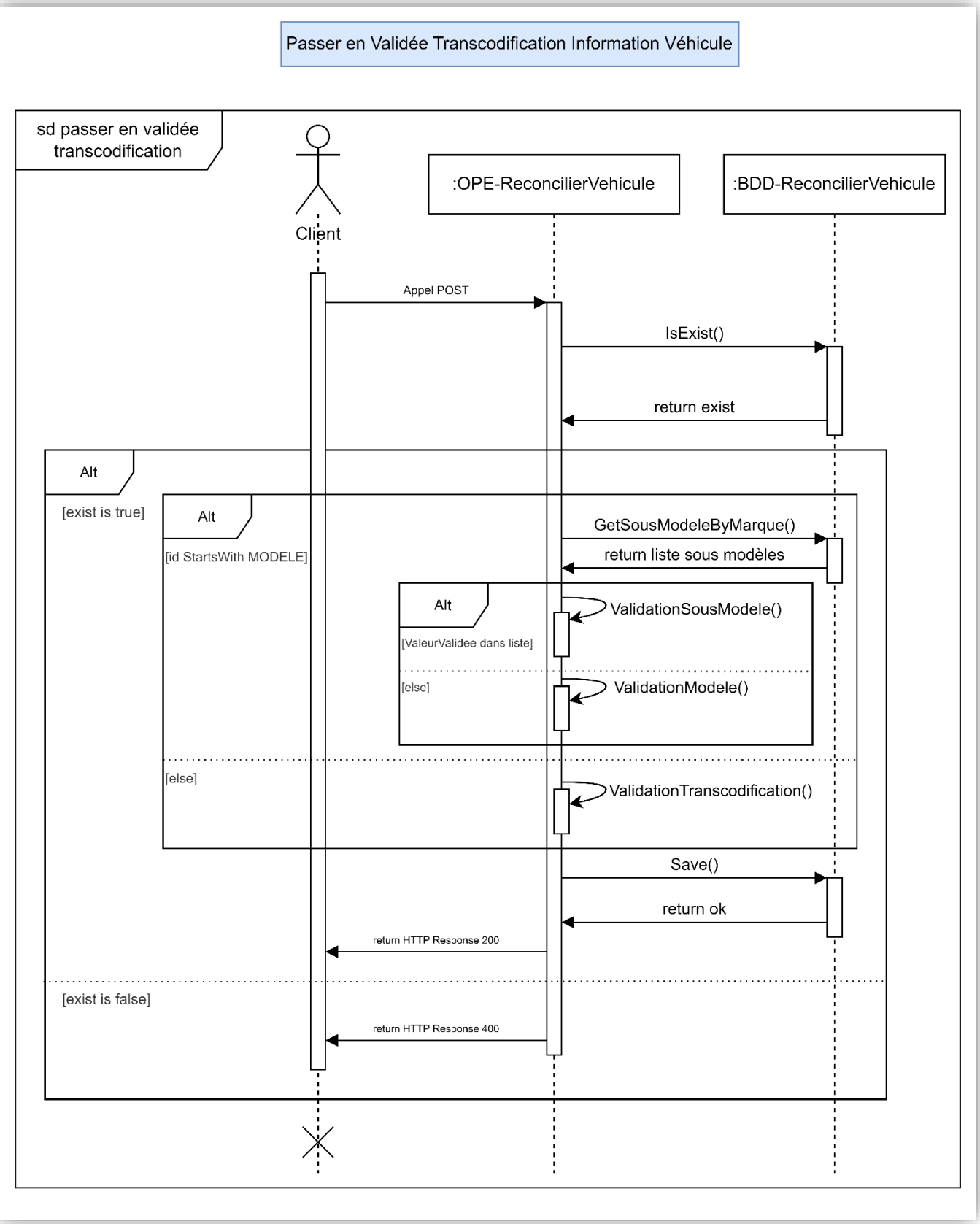
Annexe 4.4.3.a : Diagramme de séquence Consulter Transcodification



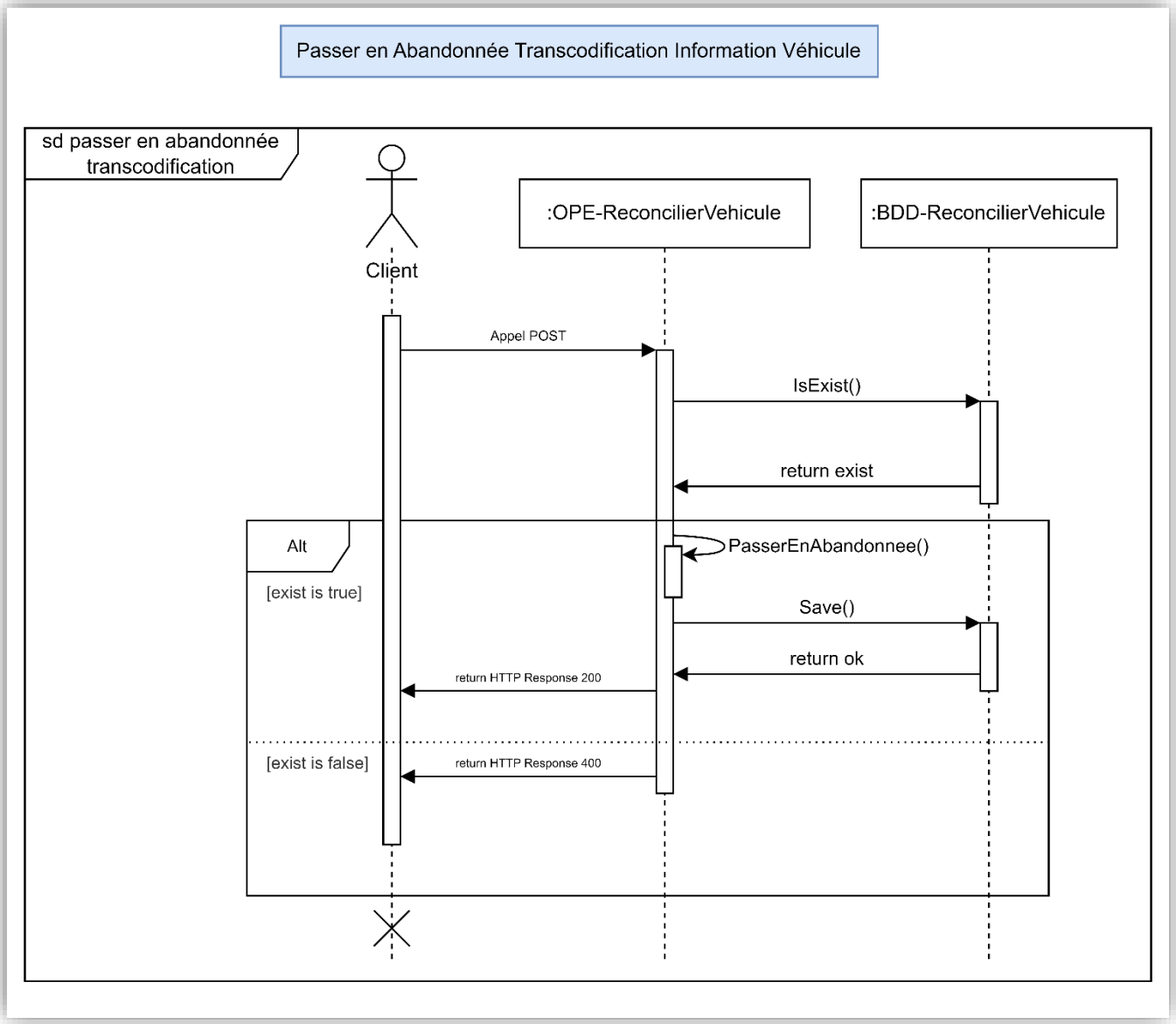
Annexe 4.4.3.b : Diagramme de séquence Créer Transcodification



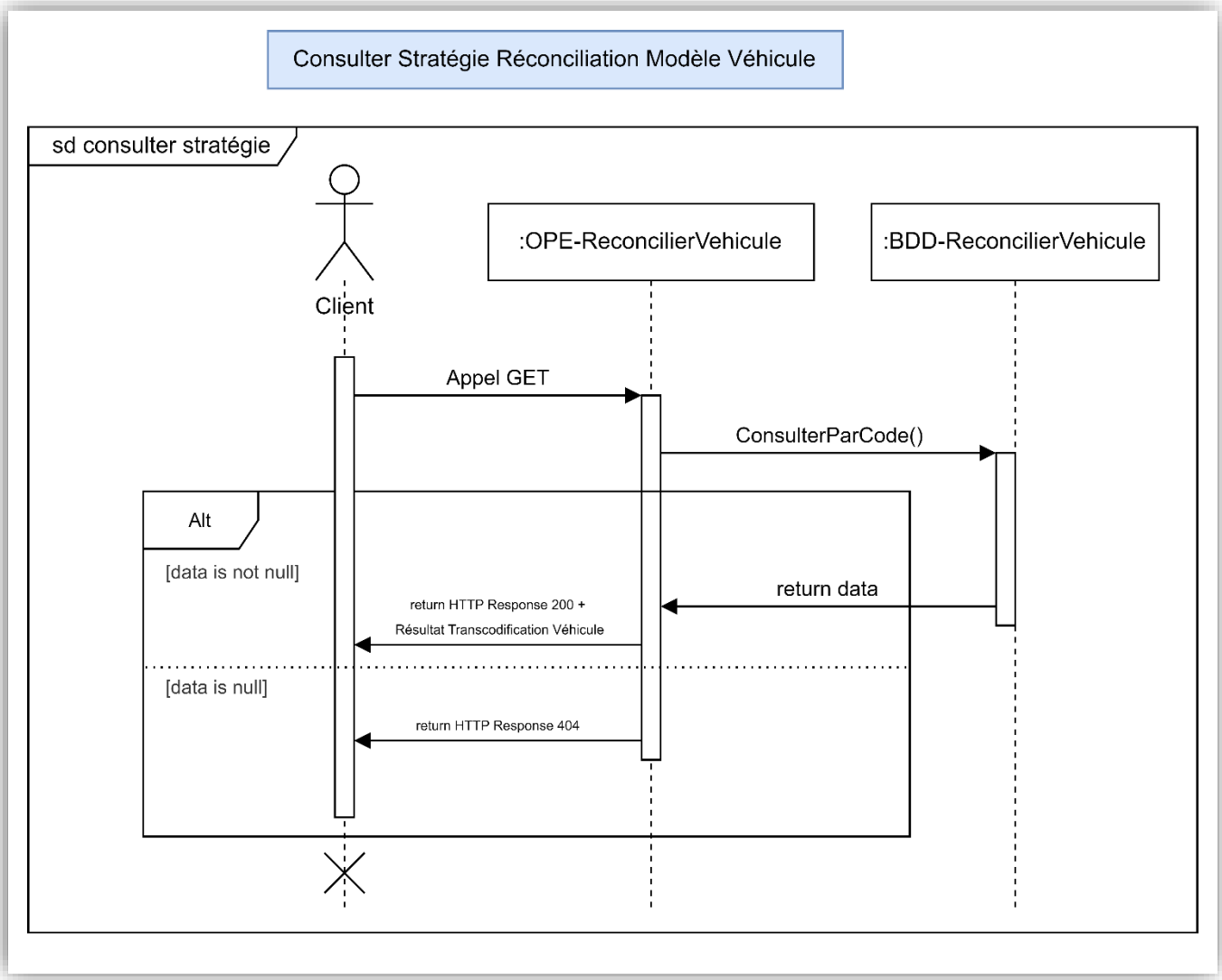
Annexe 4.4.3.c : Diagramme de séquence Passer en validée Transcodification



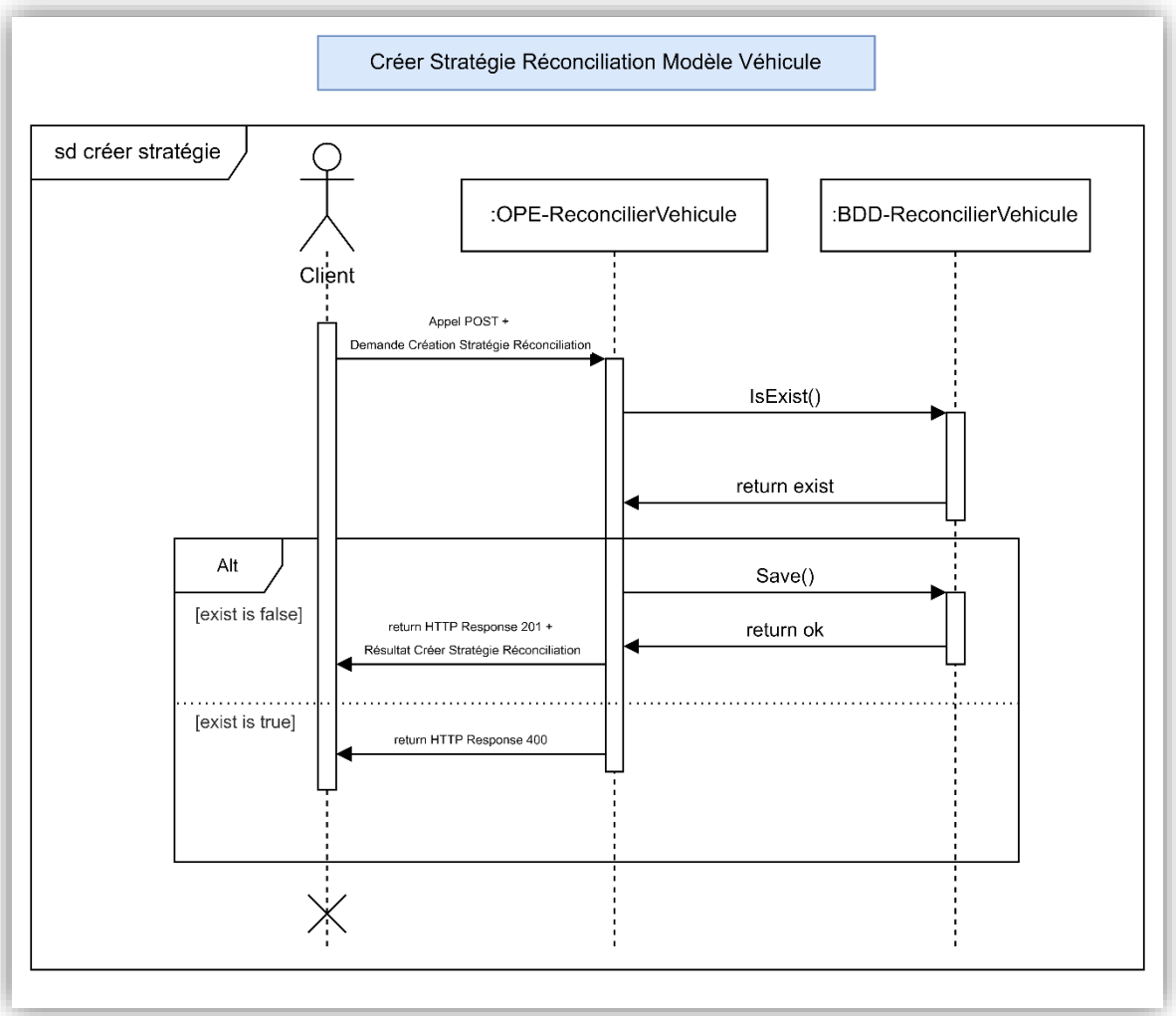
Annexe 4.4.3.d : Diagramme de séquence Passer en abandonnée Transcodification



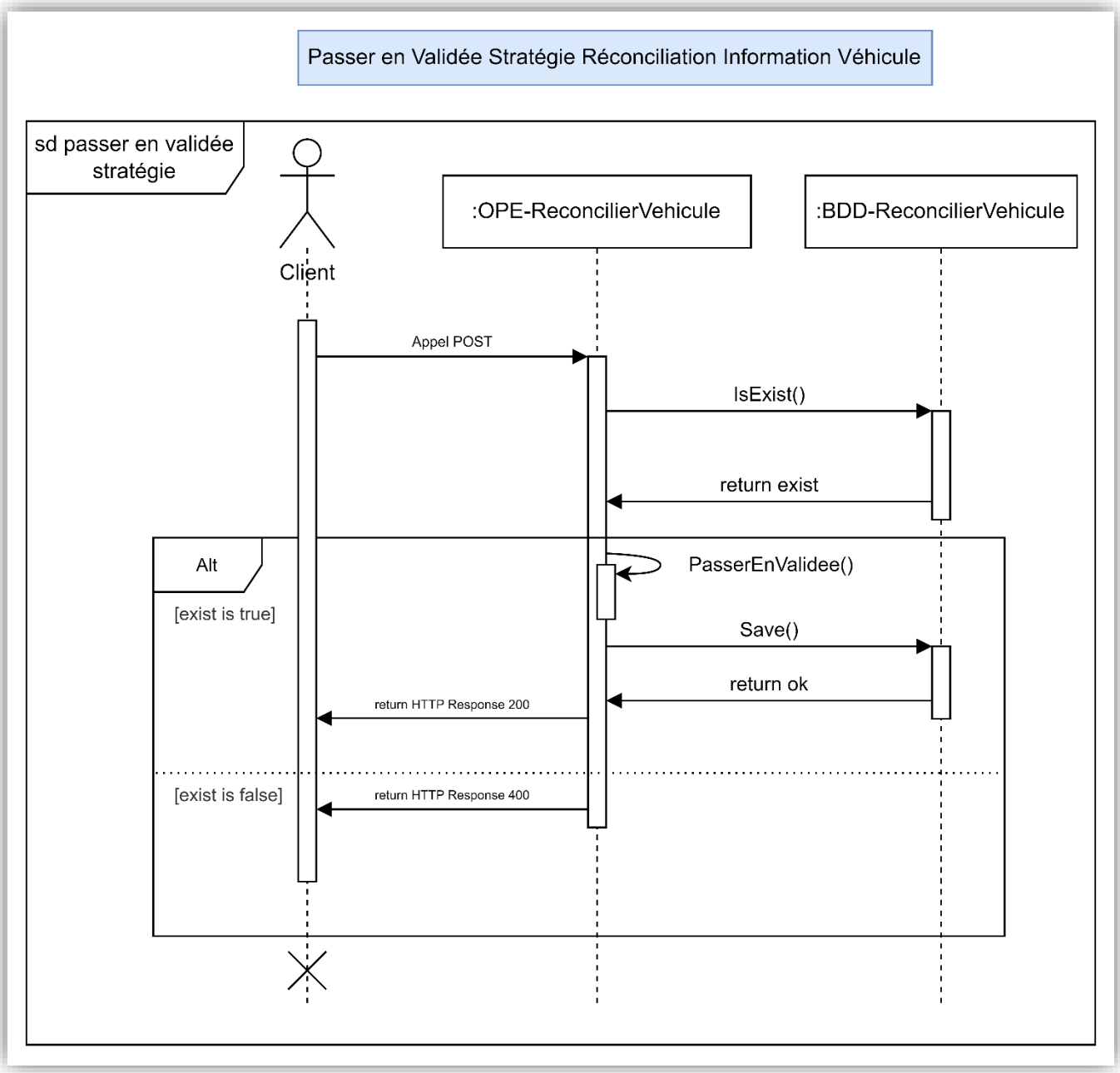
Annexe 4.4.4.a : Diagramme de séquence Consulter Stratégie Réconciliation Modèle Véhicule



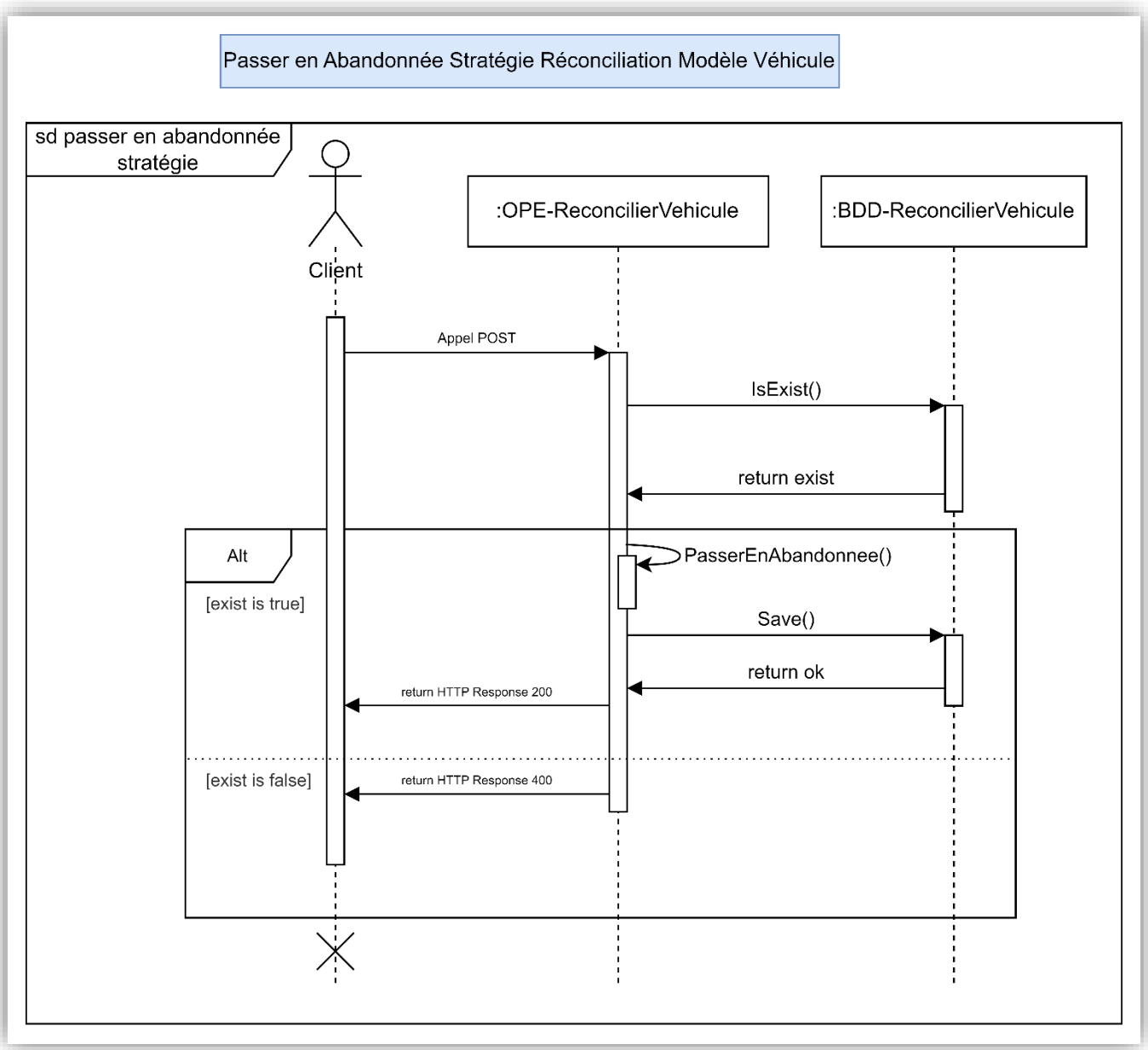
Annexe 4.4.4.b : Diagramme de séquence Créer Stratégie Réconciliation Modèle Véhicule



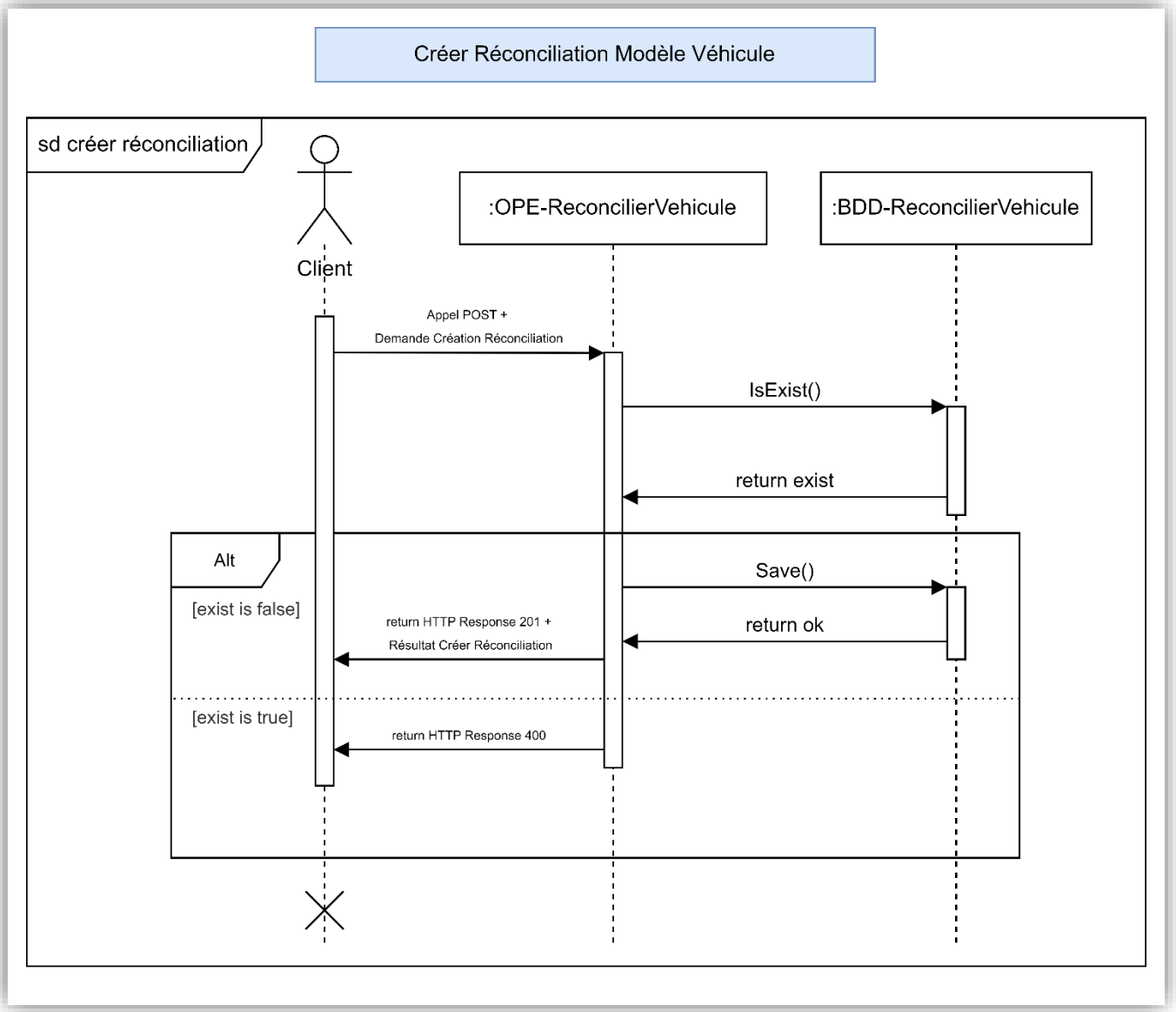
Annexe 4.4.4.c : Diagramme de séquence Passer en Validée Stratégie Réconciliation Modèle Véhicule



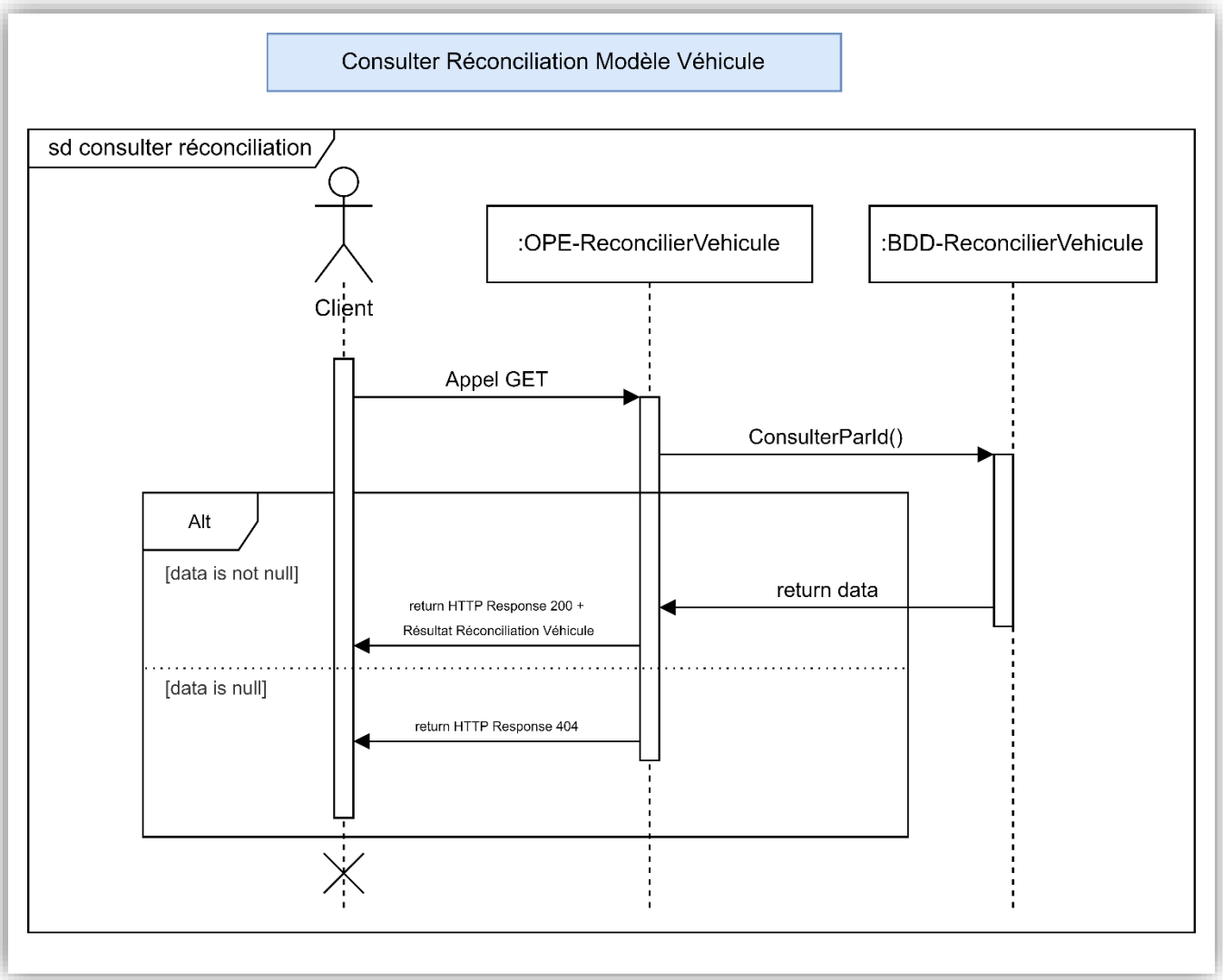
Annexe 4.4.4.d : Diagramme de séquence Passer en Abandonnée Stratégie Réconciliation Véhicule



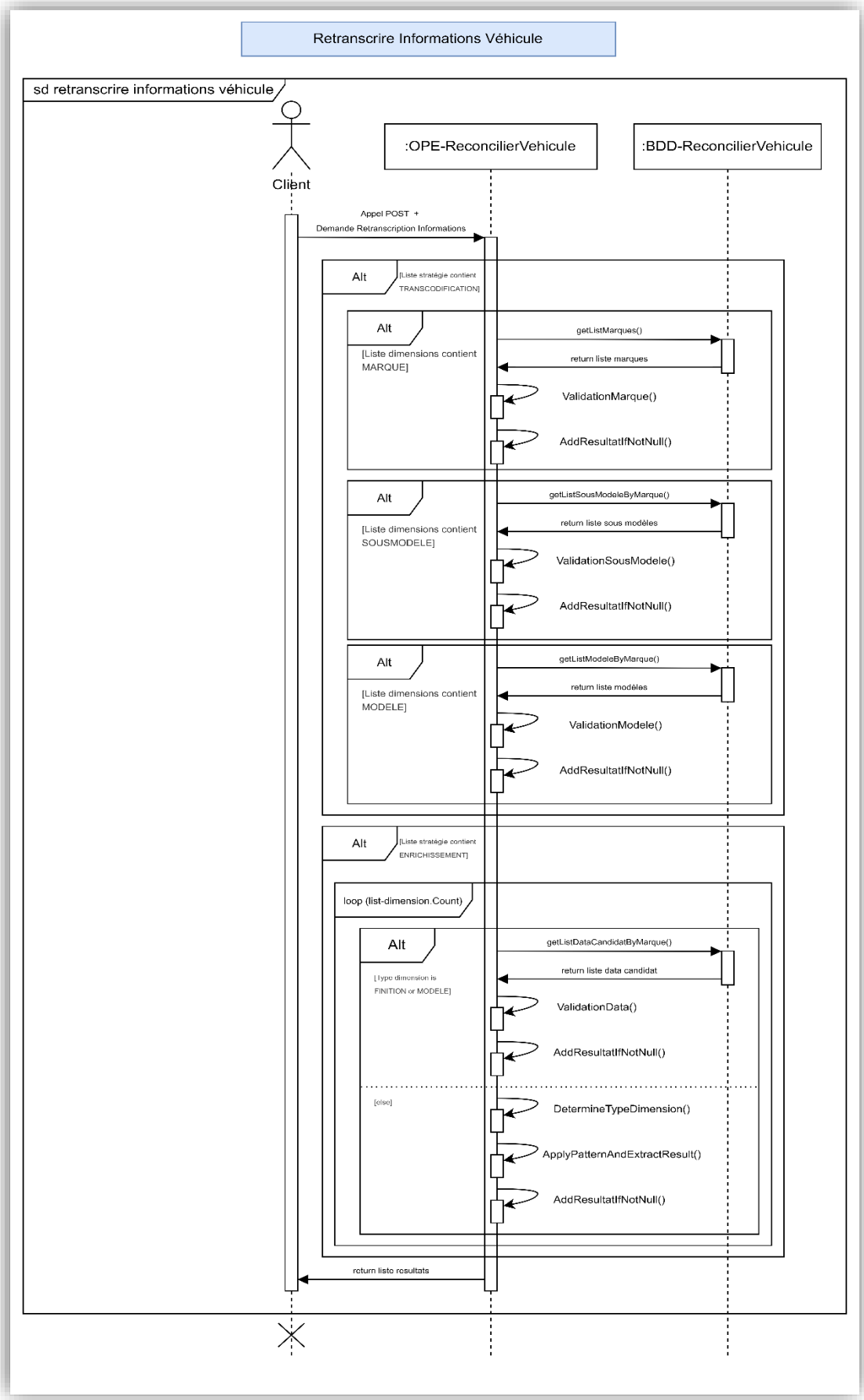
Annexe 4.4.5.a : Diagramme de séquence Créer Réconciliation Modèle Véhicule



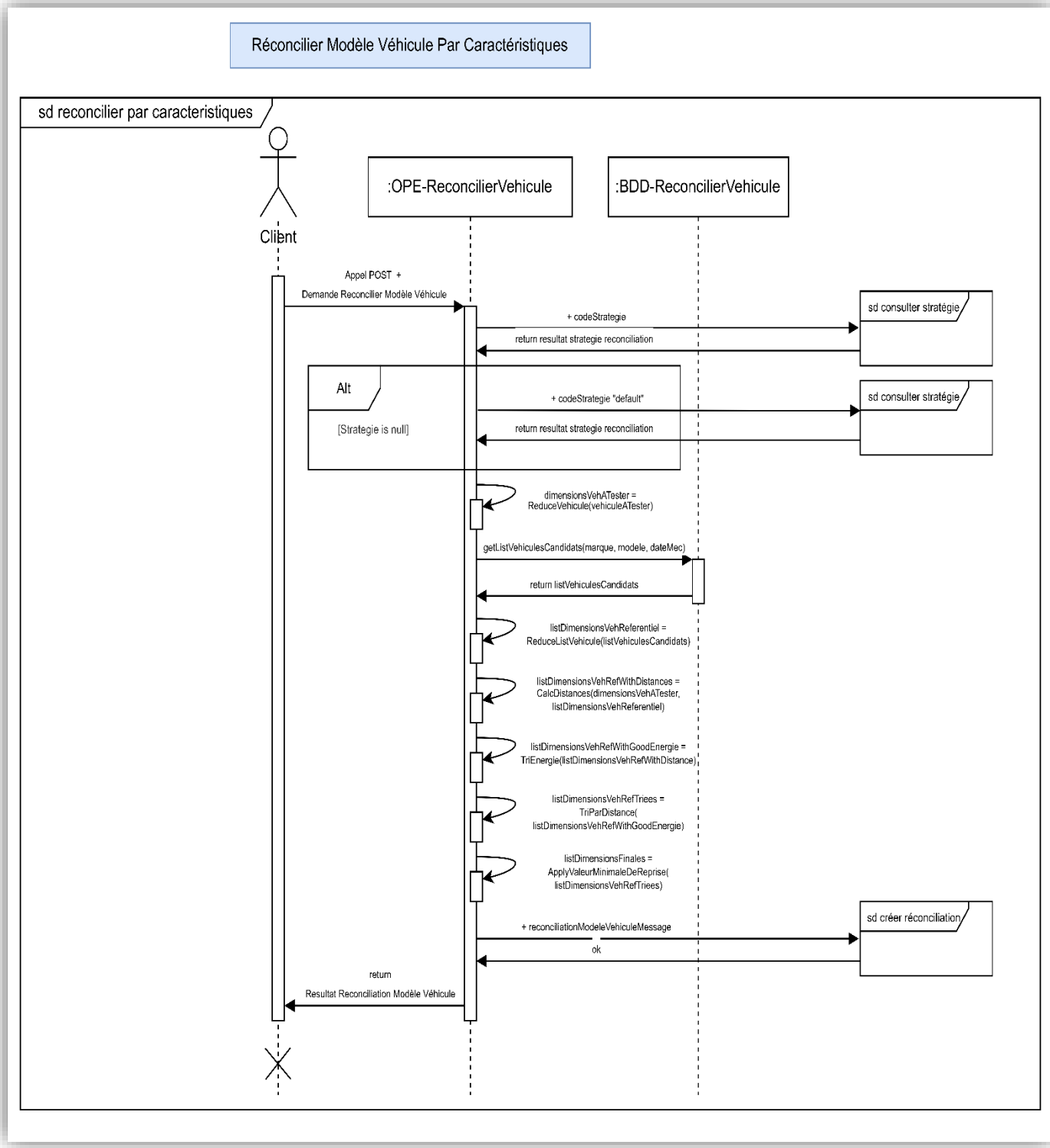
Annexe 4.4.5.b : Diagramme de séquence Consulter Réconciliation Modèle Véhicule



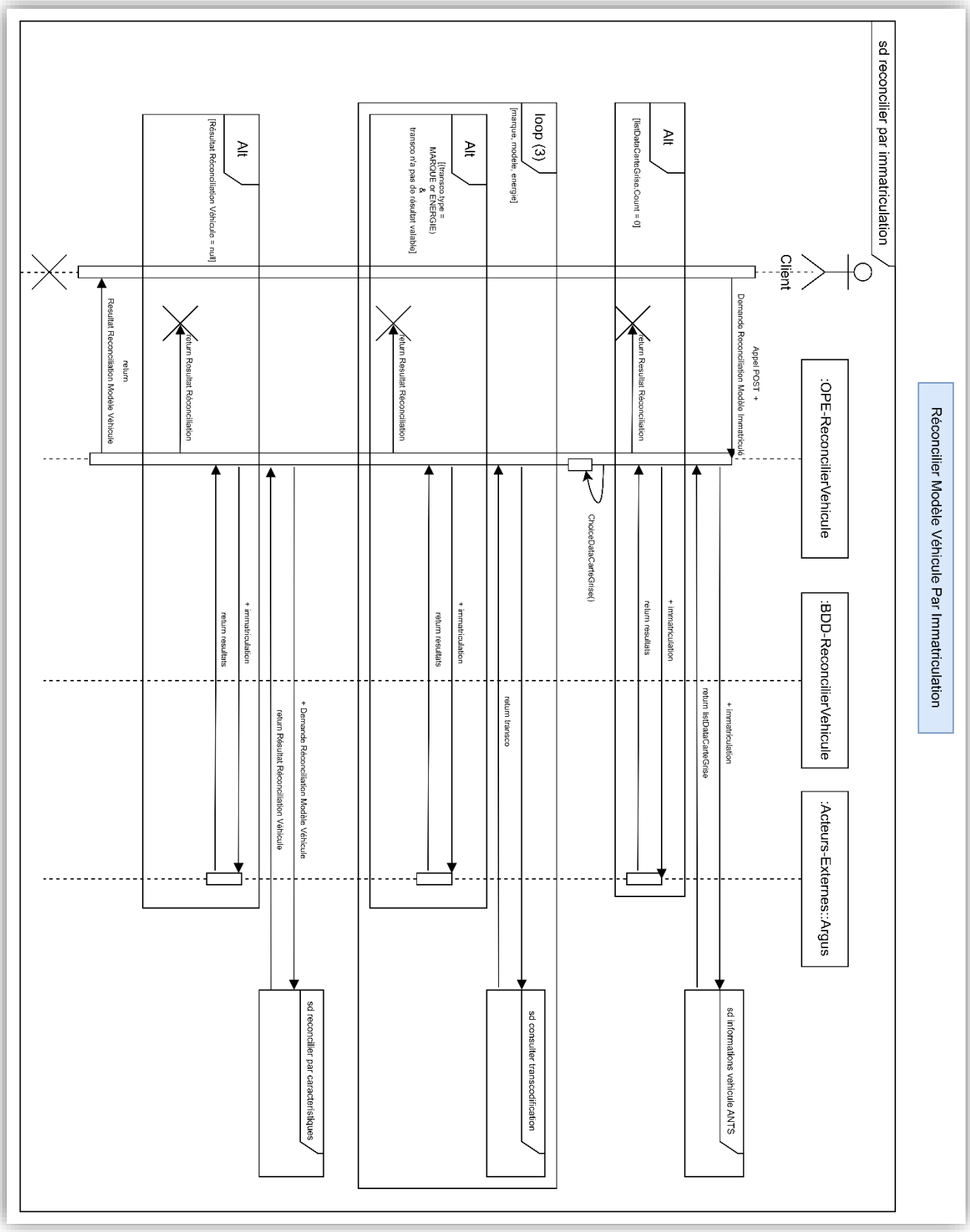
Annexe 4.4.6.a : Diagramme de séquence Retranscrire Informations Véhicule



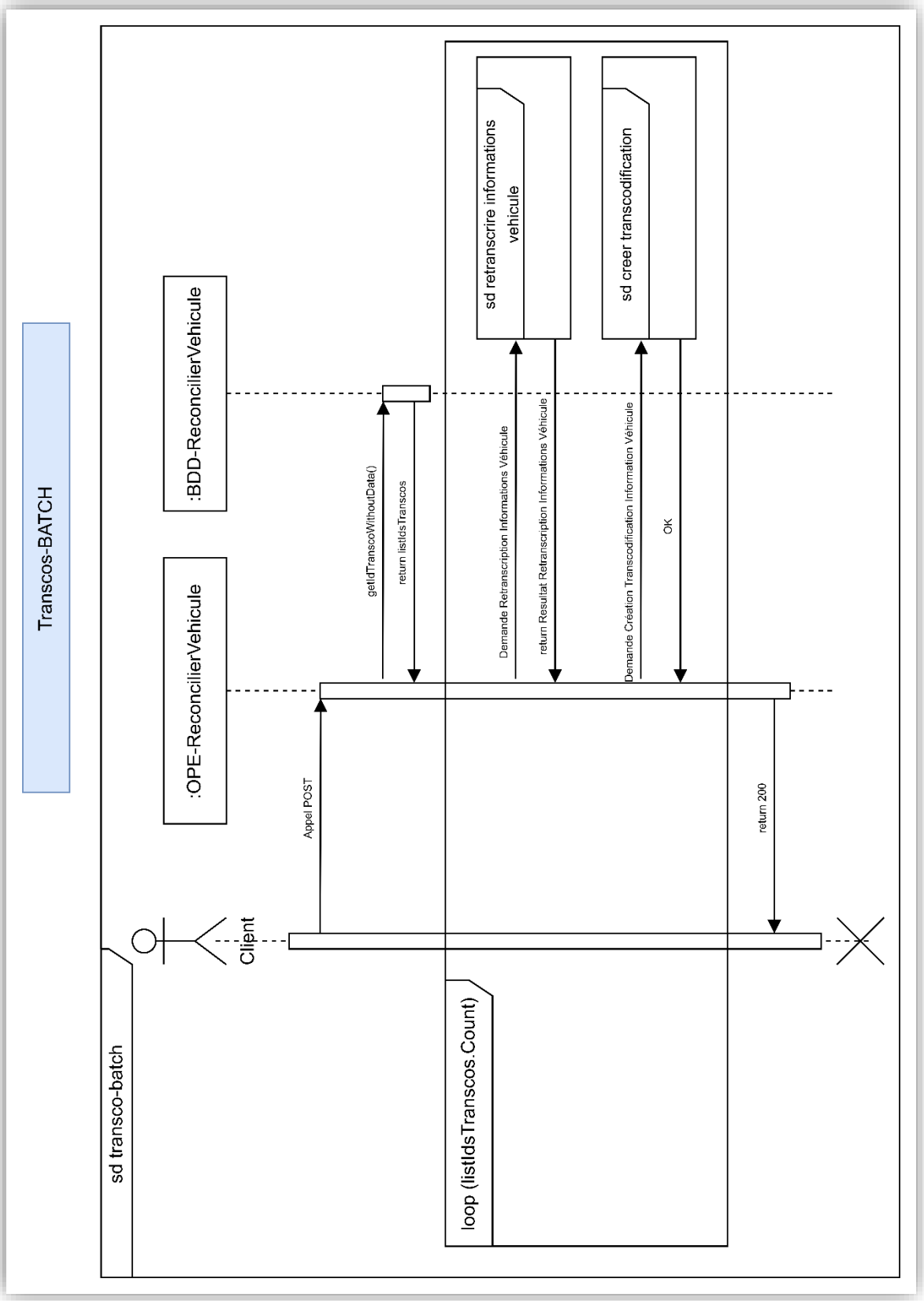
Annexe 4.4.6.b : Diagramme de séquence Réconcilier par Caractéristiques



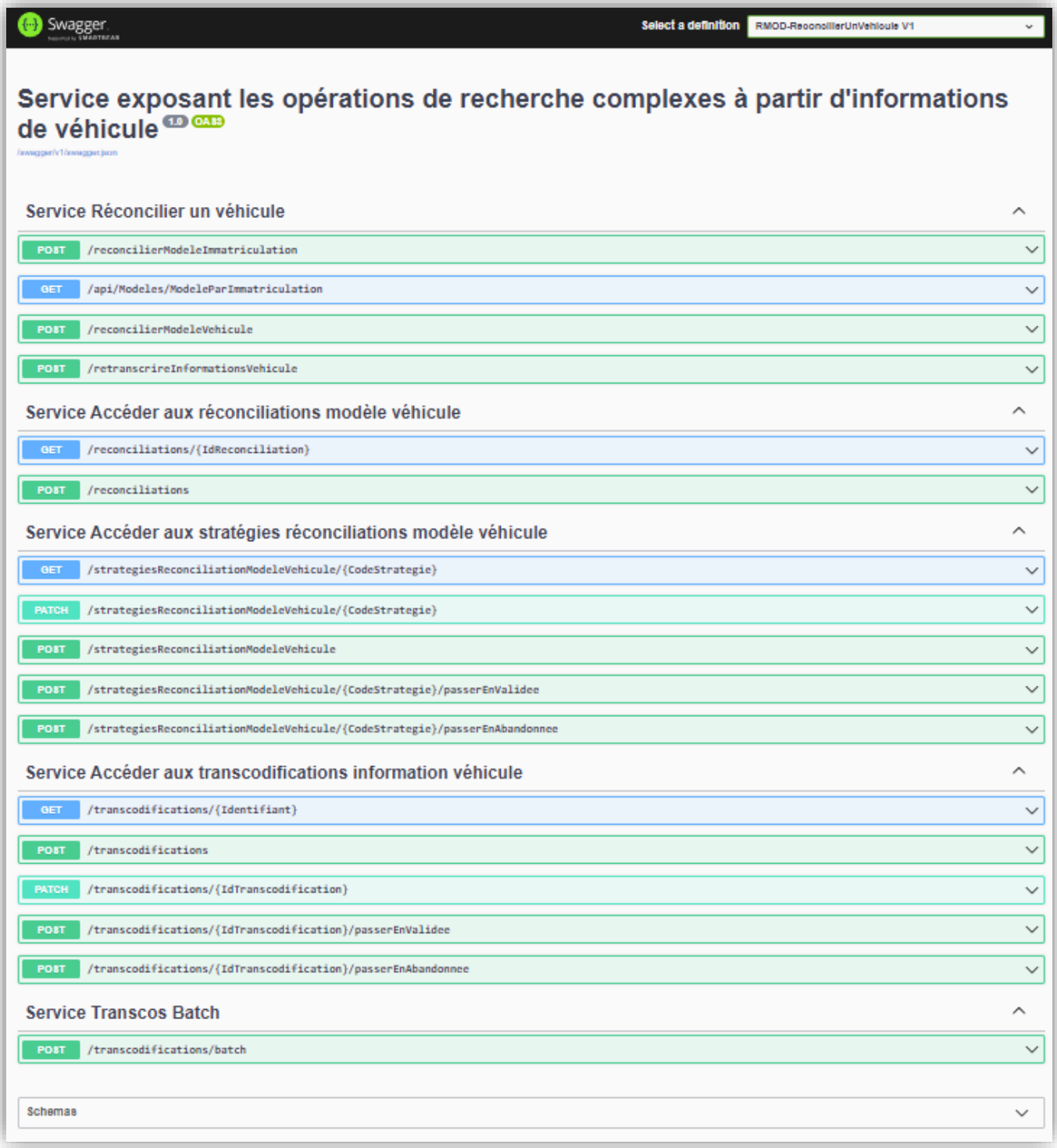
Annexe 4.4.6.c : Diagramme de séquence Réconcilier par Immatriculation



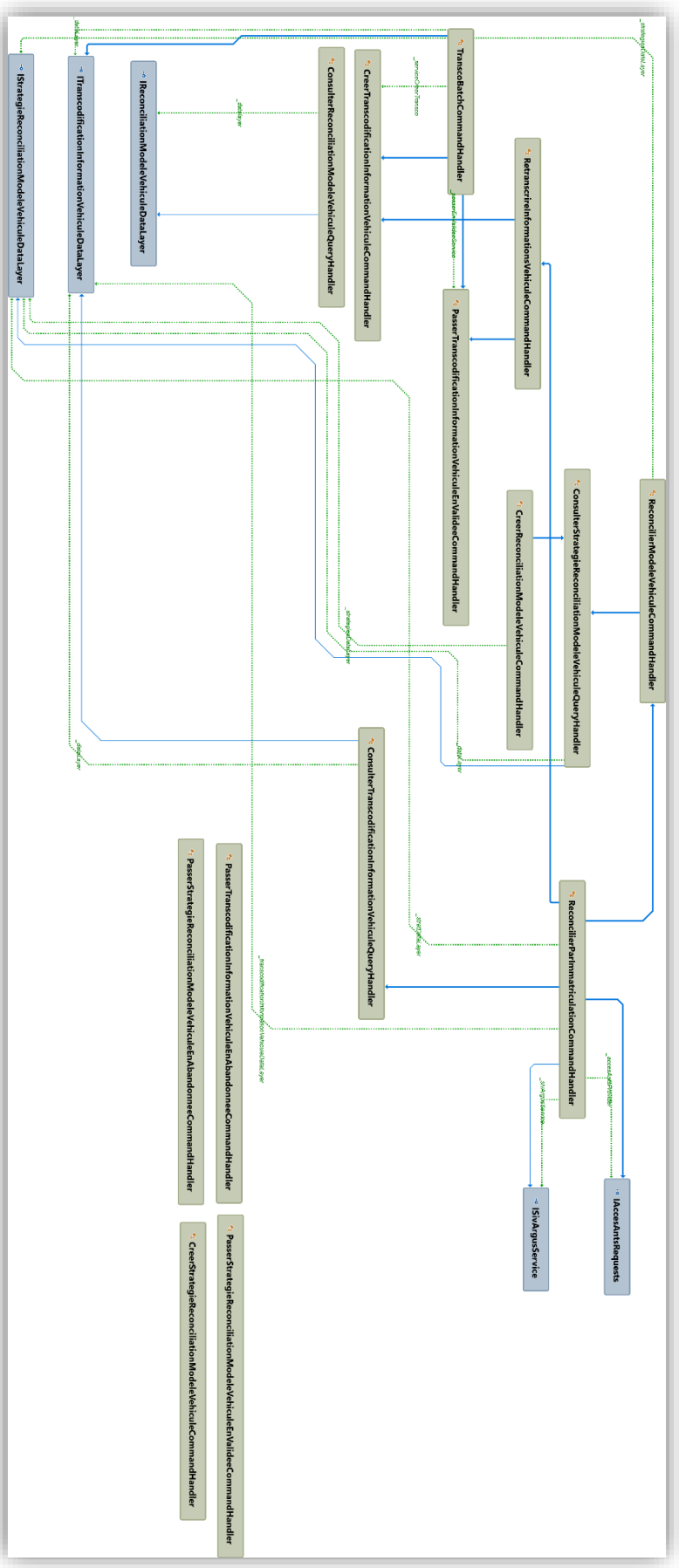
Annexe 4.4.9 : Diagramme de séquence Transcos-BATCH



Annexe 5.3.2 : Interface Swagger de l'API RMOD



Annexe 5.3.3 : Diagramme de dépendance de type des différentes opérations



Annexe 5.3.4.a : L'objet Créer Transcodification Command

```
[Category("Service Accéder aux transcodifications information véhicule")]
[Description("Permet d'enregistrer une transcodification information véhicule dans le système")]
public class CreerTranscodificationInformationVehiculeCommand :
    ICommand<CreerCommandResult<CreerTranscodificationInformationVehiculeCommandResult>>
{
    [Description("Type dimension")] public ListeDimensionVehicule CodeDimension { get; set; }

    [Description("Valeur à transcoder")] public string ValeurATranscoder { get; set; }

    [Description("Marque transcodée dans le cas d'une demande de création de transco modèle/sous-
modèle")]
    public string MarqueTranscodee { get; set; }

    [Description("Valeurs suggérées pour la transcodification")]
    public IEnumerable<string> ValeursSuggerees { get; set; }

    public string GenererId()
    {
        var id = CodeDimension == ListeDimensionVehicule.MODELE
            ? string.Concat(
                CodeDimension.ToString(), ",",
                MarqueTranscodee.ToUpper(), ",",
                ValeurATranscoder.ToUpper())
            : string.Concat(
                CodeDimension.ToString(), ",", ValeurATranscoder.ToUpper());

        return id;
    }
}
```

Annexe 5.3.4.b : L'objet Créer Transcodification Command Result

```
public class CreerTranscodificationInformationVehiculeCommandResult
{
    public string IdTranscodification { get; set; }
}
```

Annexe 5.3.4.c : L'objet Créer Transcodification Command Handler

```
public class CreerTranscodificationInformationVehiculeCommandHandler : ICommandHandler<
    CreerTranscodificationInformationVehiculeCommand,
    CreerCommandResult<CreerTranscodificationInformationVehiculeCommandResult>>
{
    private readonly IDateTimeProvider _dateTimeProvider;
    private readonly IGuidProvider _guidProvider;
    private readonly ILogger _logger;
    private readonly IAggregateRepository<TranscodificationInformationVehiculeAggregate> _repo;

    public CreerTranscodificationInformationVehiculeCommandHandler(
        IAggregateRepository<TranscodificationInformationVehiculeAggregate> repo,
        IDateTimeProvider dtProvider,
        IGuidProvider guidProvider,
        ILogger logger)
    {
        _repo = repo;
        _dateTimeProvider = dtProvider;
        _guidProvider = guidProvider;
        _logger = logger;
    }

    public async
Task<RequestResult<CreerCommandResult<CreerTranscodificationInformationVehiculeCommandResult>>>
        Handle(CreerTranscodificationInformationVehiculeCommand request)
    {
        var context = CreateContext();
        var agg = new TranscodificationInformationVehiculeAggregate(request, context, _guidProvider);

        try
        {
            await _repo.SaveAsync(agg, context);
        }
        catch (OptimisticConcurrencyException optimisticException)
        {
            _logger.LogError(optimisticException, $"Impossible de créer la transco d'identifiant
{request.GenererId()} car elle existe déjà");

            return ObjetMetierErrorHelper.Validation.Add(
                    ErrorConstants.ERR_DUPLICATEOBJECT,
                    "La transco existe déjà",
                    nameof(CreerTranscodificationInformationVehiculeCommandResult),
                    request)

.BuildRequestResult<CreerCommandResult<CreerTranscodificationInformationVehiculeCommandResult>>();
        }
        catch (Exception e)
        {
            _logger.LogError(e, $"Impossible de créer la transco d'identifiant {request.GenererId()}");

            throw;
        }

        return RequestResult.Success(
            new CreerCommandResult<CreerTranscodificationInformationVehiculeCommandResult>
            {
                Data = new CreerTranscodificationInformationVehiculeCommandResult
                {
                    IdTranscodification = agg.GetState().Id
                }
            });
    }

    private AggregateChangeContext CreateContext()
    {
        return new AggregateChangeContext
        {
            ActionDate = _dateTimeProvider.NowOffset,
            Application = MessageContext.Current.GetNomServiceInital(),
            CorrelationId = MessageContext.Current.GetIdCorrelation(),
            User = MessageContext.Current.GetUtilisateur()
        };
    }
}
```

Annexe 5.3.4.d : L'objet Créer Transcodification Aggregate

```
public class TranscodificationInformationVehiculeAggregate :
AggregateRoot<TranscodificationInformationVehiculeMessage>
{
    public TranscodificationInformationVehiculeAggregate(TranscodificationInformationVehiculeMessage
state) :
        base(state)
    {
    }

    public TranscodificationInformationVehiculeAggregate(CreerTranscodificationInformationVehiculeCommand
request,
        AggregateChangeContext context, IGuidProvider guidProvider)
    {
        State = new TranscodificationInformationVehiculeMessage
        {
            Id = request.GenererId(),
            ValeurATranscoder = request.ValeurATranscoder,
            ValeursSuggerees = request.ValeursSuggerees,
            Dimension = request.CodeDimension,
            AutresIdentifiants = null,
            CycleDeVie = new CycleDeVie<StatutTranscodificationInformationVehicule>
            {
                Statut = StatutTranscodificationInformationVehicule.INITIALISEE,
                CodeApplicationOrigineChangement = context.Application,
                CodeUtilisateur = context.User,
                DateChangement = context.ActionDate
            },
            DonneesAudit = new Audit
            {
                CodeApplicationCreation = context.Application,
                CodeUtilisateurCreation = context.User,
                DateCreation = context.ActionDate
            }
        };
    }
}
```

Annexe 5.3.5.a : Récupération de la liste des sous-modèles par marque

```
public async Task<List<ModeleSousModele>> GetListModeleSousModeleByMarqueAsync(string marqueToFind)
{
    using var dbContext = RecupererDbContext();

    var query = from version in dbContext.Versions
                join sousmodele in dbContext.SousModeles on version.IdSousmodele equals sousmodele.Id
                join modele in dbContext.Modeles on version.IdModele equals modele.Id
                join marque in dbContext.Marques on version.IdMarque equals marque.Id
                where marque.Libelle == marqueToFind
                select new ModeleSousModele
                {
                    Modele = modele.Libelle,
                    SousModele = sousmodele.Libelle
                };

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

Annexe 5.3.5.b : L'objet Passer transcodification en validée Command Handler

```
public class
    PasserTranscodificationInformationVehiculeEnValideeCommandHandler : ICommandHandler<
        PasserTranscodificationInformationVehiculeEnValideeCommand>
{
    private readonly IDateTimeProvider _dateTimeProvider;
    private readonly IAggregateRepository<TranscodificationInformationVehiculeAggregate> _transcoRepo;
    private readonly ICatalogueConstructeurAdapter _catalogueConstructeur;

    public PasserTranscodificationInformationVehiculeEnValideeCommandHandler(
        IAggregateRepository<TranscodificationInformationVehiculeAggregate> transcoRepo,
        IDateTimeProvider dateTimeProvider,
        ICatalogueConstructeurAdapter catalogueConstructeur)
    {
        _transcoRepo = transcoRepo;
        _dateTimeProvider = dateTimeProvider;
        _catalogueConstructeur = catalogueConstructeur;
    }

    public async Task<RequestResult<Nothing>>
Handle(PasserTranscodificationInformationVehiculeEnValideeCommand request)
    {
        var transcoExistante = await _transcoRepo.LoadAsync(request.IdTranscodification);
        if (transcoExistante == null)
        {
            return ObjetMetierErrorHelper.NotFound
                .Set($"Aucune transcodification n'a été trouvé pour le code
'{request.IdTranscodification}'",
                    nameof(request.IdTranscodification), request.IdTranscodification)
                .BuildRequestResult<Nothing>();
        }

        var context = new AggregateChangeContext
        {
            Application = MessageContext.Current.GetNomServiceInital(),
            User = MessageContext.Current.GetUtilisateur(),
            ActionDate = MessageContext.Current.DateOffset,
            CorrelationId = MessageContext.Current.GetIdCorrelation()
        };

        if (transcoExistante.GetState().Dimension ==
RMOD_ReconcilierUnVehicule.Messages.Enums.ListeDimensionVehicule.SOUSMODELE)
        {
            transcoExistante.ReinitializeModeleDimension();
        }

        if (transcoExistante.GetState().Dimension ==
RMOD_ReconcilierUnVehicule.Messages.Enums.ListeDimensionVehicule.MODELE)
        {
            transcoExistante = await CheckIfModeleSousModele(transcoExistante, request);
        }

        transcoExistante.PasserEnValidee(request, context, _dateTimeProvider);

        await _transcoRepo.SaveAsync(transcoExistante, context);

        return RequestResult.Nothing();
    }

    private async Task<TranscodificationInformationVehiculeAggregate>
CheckIfModeleSousModele(TranscodificationInformationVehiculeAggregate transcoExistante,
PasserTranscodificationInformationVehiculeEnValideeCommand request)
    {
        var marque = transcoExistante.GetState().Id.Split(',')[1];

        if (marque == request.ValeurValidee.ToUpper())
        {
            return transcoExistante;
        }
        else
        {
            var listModeleSousModele = await
_catalogueConstructeur.GetListModeleSousModeleByMarqueAsync(marque);

            foreach (var item in listModeleSousModele)
            {
                item.SousModele = item.SousModele.ToUpper();
                item.Modele = item.Modele.ToUpper();
            }

            var checkIfContains = listModeleSousModele.Where(p => string.Equals(p.SousModele,
request.ValeurValidee.ToUpper()));

            if (checkIfContains.Any())
            {
                if (checkIfContains?.FirstOrDefault()?.Modele !=
checkIfContains?.FirstOrDefault()?.SousModele)
                {
                    transcoExistante.TransformToSousModele();
                }
            }

            return transcoExistante;
        }

    }
}
```

Annexe 5.3.5.c : Méthode Passer En Validée de la classe Transcodification Aggregate

```
public void PasserEnValidee(PasserTranscodificationInformationVehiculeEnValideeCommand request,
    AggregateChangeContext context, IDateTimeProvider dateTimeProvider)
{
    var transco = State;
    transco.CycleDeVie.Statut = StatutTranscodificationInformationVehicule.VALIDEE;
    transco.Validation = new Validation { ValeurValidee = request.ValeurValidee };

    transco.DonneesAudit = new Audit
    {
        CodeApplicationCreation = State.DonneesAudit?.CodeApplicationCreation,
        CodeUtilisateurCreation = State.DonneesAudit?.CodeUtilisateurCreation,
        DateCreation = State.DonneesAudit?.DateCreation,
        CodeApplicationModification = context.Application,
        CodeUtilisateurModification = context.User,
        DateModification = dateTimeProvider.Now
    };
}

public void TransformToSousModele()
{
    State.Dimension = ListeDimensionVehicule.SOUSMODELE;
}

public void ReinitializeModeleDimension()
{
    State.Dimension = ListeDimensionVehicule.MODELE;
}
```

Annexe 5.3.6.a : La classe Consulter Transcodifications Query Handler

```
public class ConsulterTranscodificationInformationVehiculeQueryHandler : IQueryHandler<
    ConsulterTranscodificationInformationVehiculeQuery,
ConsulterTranscodificationInformationVehiculeQueryResult>
{
    private readonly ITranscodificationInformationVehiculeDataLayer _dataLayer;

    public ConsulterTranscodificationInformationVehiculeQueryHandler(
        ITranscodificationInformationVehiculeDataLayer dataLayer)
    {
        _dataLayer = dataLayer;
    }

    public async Task<RequestResult<ConsulterTranscodificationInformationVehiculeQueryResult>> Handle(
        ConsulterTranscodificationInformationVehiculeQuery request)
    {
        var message = await _dataLayer.ConsulterParId(request.Identifiant);

        if (message != null)
        {
            return RequestResult.Success(
                new ConsulterTranscodificationInformationVehiculeQueryResult { Data = message });
        }

        return ObjetMetierErrorHelper.NotFound
            .Set($"Aucune transcodification n'a été trouvée pour l'identifiant '{request.Identifiant}'",
                nameof(request.Identifiant), request.Identifiant)
            .BuildRequestResult<ConsulterTranscodificationInformationVehiculeQueryResult>();
    }
}
```

Annexe 5.3.6.b : Méthode Récupérer par Id de la classe Transcodification Aggregate

```
public async Task<TranscodificationInformationVehiculeMessage> ConsulterParId(string id)
{
    await using var con = new NpgsqlConnection(_connectionString);

    var serializedObject = await con.QuerySingleOrDefaultAsync<string>(
        "select doc from transcodification_information_vehicule where id = :Id limit 1",
        new { Id = id });

    if (serializedObject is null)
    {
        return null;
    }

    return JsonConvert.DeserializeObject<TranscodificationInformationVehiculeMessage>(
        serializedObject,
        _serializerSettings);
}
```

Annexe 5.3.7 : Classe passer transcodification en abandonnée

```
public class
    PasserTranscodificationInformationVehiculeEnAbandonneeCommandHandler : ICommandHandler<
        PasserTranscodificationInformationVehiculeEnAbandonneeCommand>
{
    private readonly IDateTimeProvider _dateTimeProvider;
    private readonly IAggregateRepository<TranscodificationInformationVehiculeAggregate> _transcoRepo;

    public PasserTranscodificationInformationVehiculeEnAbandonneeCommandHandler(
        IAggregateRepository<TranscodificationInformationVehiculeAggregate> transcoRepo,
        IDateTimeProvider dateTimeProvider)
    {
        _transcoRepo = transcoRepo;
        _dateTimeProvider = dateTimeProvider;
    }

    public async Task<RequestResult<Nothing>> Handle(
        PasserTranscodificationInformationVehiculeEnAbandonneeCommand request)
    {
        var transcoExistante = await _transcoRepo.LoadAsync(request.IdTranscodification);
        if (transcoExistante == null)
        {
            return ObjetMetierErrorHelper.NotFound
                .Set($"Aucune transcodification n'a été trouvé pour le code
'{request.IdTranscodification}'",
                    nameof(request.IdTranscodification), request.IdTranscodification)
                .BuildRequestResult<Nothing>();
        }

        if (transcoExistante.GetState().CycleDeVie.Statut ==
StatutTranscodificationInformationVehicule.ABANDONNEE)
        {
            return ObjetMetierErrorHelper
                .Validation.Add(
                    ErrorConstants.ERR_INVALIDSTATE,
                    "La transcodification est déjà au statut abandonnée",
                    nameof(RetranscrireInformationsVehiculeCommand),
                    request)
                .BuildRequestResult<Nothing>();
        }

        var context = new AggregateChangeContext
        {
            Application = MessageContext.Current.GetNomServiceInital(),
            User = MessageContext.Current.GetUtilisateur(),
            ActionDate = MessageContext.Current.DateOffset,
            CorrelationId = MessageContext.Current.GetIdCorrelation()
        };

        transcoExistante.PasserEnAbandonnee(context, _dateTimeProvider);

        await _transcoRepo.SaveAsync(transcoExistante, context);

        return RequestResult.Nothing();
    }
}
```

Annexe 5.3.8.a : Méthode Reduce de la classe InfosVehicule

```
public class InfosVehicule : IInfosVehicule
{
    public int? IdArgus { get; set; }
    public string? Marque { get; set; }
    public string? Modele { get; set; }
    public string? SousModele { get; set; }
    public DateTime? DatePremiereMec { get; set; }
    public DateTime? DateDebut { get; set; }
    public DateTime? DateFin { get; set; }
    public string? Energie { get; set; }
    public string? BoiteVitesse { get; set; }
    public string? TypeBoiteVitesse { get; set; }
    public float? PuissanceFiscale { get; set; }
    public float? PuissanceDin { get; set; }
    public int? NombreDePlaces { get; set; }
    public int? NombreDePortes { get; set; }
    public float? EmissionCo2 { get; set; }
    public float? Cylindree { get; set; }
    public float? Poids { get; set; }
    public float? Consommation { get; set; }
    public string? Carrosserie { get; set; }
    public string? Generation { get; set; }
    public string? Phase { get; set; }
    public string? Version { get; set; }
    public float? PrixCatalogue { get; set; }
    public string? Finition { get; set; }
    public DateTime? DateDebutPrixCatalogue { get; set; }
    public DateTime? DateFinPrixCatalogue { get; set; }


    /// <summary>
    ///     Change les caractéristiques des véhicules en liste de dimensions
    /// </summary>
    /// <param name="strategy">La stratégie de la réconciliation</param>
    /// <returns>Liste de dimensions</returns>
    public IListDimensionsVehicule? Reduce(IStrategieReconciliationModeleVehicule strategy, bool
needMarqueModeleDateMec)
    {
        var listDimensionsVehicle = new ListDimensionsVehicule
        {
            InfosVehicule = this,
            Dimensions = new List<IDimension>()
        };

        var infosVehicule = GetType().GetProperties(BindingFlags.Instance | BindingFlags.Public);

        foreach (var property in infosVehicule)
        {
            var name = property.Name;
            var value = property.GetValue(this);

            if (needMarqueModeleDateMec && IsMarqueModeleDatePremiereMecNotPresents(name, value))
            {
                return null;
            }

            if (value != null)
            {
                var dimensionFactory = new DimensionFactory(name, value, strategy);
                var dimension = dimensionFactory.Build();

                if (dimension != null)
                {
                    listDimensionsVehicle.Dimensions.Add(dimension);
                }
            }
        }

        return listDimensionsVehicle;
    }

    private static bool IsMarqueModeleDatePremiereMecNotPresents(string name, object value)
    {
        switch (name)
        {
            case "Marque":
            case "Modele":
            case "DatePremiereMec":
                {
                    switch (value)
                    {
                        case null:
                            return true;
                    }

                    break;
                }
        }

        return false;
    }
}
```

Annexe 5.3.8.b : La DimensionFactory

```
public class DimensionFactory
{
    private readonly string _nameProperty;
    private readonly IStrategieReconciliationModeleVehicule _strategy;
    private object _value;

    public DimensionFactory(string nameProperty, object value, IStrategieReconciliationModeleVehicule
strategy)
    {
        _nameProperty = nameProperty;
        _value = value;
        _strategy = strategy;
    }

    public IDimension? Build()
    {
        ListDimensionsVehicules typeDimension;

        switch (_nameProperty)
        {
            case "Marque":
                typeDimension = ListDimensionsVehicules.MARQUE;
                _value = _value.ToString().ToUpper();
                break;
            case "Modele":
                typeDimension = ListDimensionsVehicules.MODELE;
                _value = _value.ToString().ToUpper();
                break;
            case "SousModele":
                typeDimension = ListDimensionsVehicules.SOUSMODELE;
                _value = _value.ToString().ToUpper();
                break;
            case "DatePremiereMec":
                typeDimension = ListDimensionsVehicules.DATE_PREMIERE_MEC;
                break;

            /// Ect...

            case "Finition":
                typeDimension = ListDimensionsVehicules.FINITION;
                _value = _value.ToString().ToUpper();
                break;
            default:
                return null;
        }

        if (_strategy != null)
        {
            var ponderation = _strategy.DimensionsPonderees
                .Where(t => t.CodeDimension == typeDimension)
                .Select(x => x.Ponderation)
                .FirstOrDefault();

            return new Dimension(_value, typeDimension, ponderation);
        }

        return new Dimension(_value, typeDimension);
    }
}
```

Annexe 5.3.8.c : La classe Dimension

```
public class Dimension : IDimension
{
    public Dimension(object value, ListDimensionsVehicules type, float ponderation)
    {
        Value = value;
        Type = type;
        Ponderation = ponderation;
    }

    public Dimension(object value, ListDimensionsVehicules type)
    {
        Value = value;
        Type = type;
    }

    public object Value { get; set; }
    public float Ponderation { get; set; }
    public float? DistanceDimension { get; set; }
    public ListDimensionsVehicules Type { get; set; }

    public void CalculerDistance(IDimension dimensionTest)
    {
        var calcDistanceFactory = new CalculDistanceFactory(this, dimensionTest);
        var calcDistanceType = calcDistanceFactory.Build();
        DistanceDimension = calcDistanceType?.CalcDistance();
    }
}
```

Annexe 5.3.10.a : Le service de transcodification automatique

```
public class TranscoderVehiculeService : ITranscoderVehiculeService
{
    private readonly ITranscoProcess _transcoProcess;

    public TranscoderVehiculeService(ITranscoProcess transcoProcess)
    {
        _transcoProcess = transcoProcess;
    }

    public async Task<SuggestTranscos> CreerTranscosMarqueAutomatique(string marque)
    {
        return await _transcoProcess.CreerTranscoMarqueAutomatique(marque);
    }

    public async Task<SuggestTranscos> CreerTranscoModeleAutomatique(string marque, string modele)
    {
        return await _transcoProcess.CreerTranscoModeleAutomatique(marque, modele);
    }

    public async Task<SuggestTranscos> CreerTranscoSousModeleAutomatique(string marque, string modele)
    {
        return await _transcoProcess.CreerTranscoSousModeleAutomatique(marque, modele);
    }
}
```

Annexe 5.3.10.b : Le middleware TranscoProcess

```
public class TranscoProcess : ITranscoProcess
{
    private const string ClefCacheMarques = "marques";
    private readonly IMemoryCache _cache;
    private readonly ICatalogueConstructeurAdapter _catalogueConstructeurQueryHandler;

    public TranscoProcess(
        ICatalogueConstructeurAdapter catalogueConstructeurQueryHandlerHandler,
        IMemoryCache cache)
    {
        _catalogueConstructeurQueryHandler = catalogueConstructeurQueryHandlerHandler;
        _cache = cache;
    }

    public List<IDimension> DetermineModeleSouModele(List<IDimension> listDimVeh)
    {
        // TODO process pour determiner si dans le champ modele c'est un modele ou un sous modele
        throw new NotImplementedException();
    }

    public async Task<SuggestTranscos> CreerTranscoMarqueAutomatique(string marque)
    {
        if (_cache.TryGetValue(ClefCacheMarques, out List<DatasCatalogueConstructeur> marques) is
false)
        {
            marques = await _catalogueConstructeurQueryHandler.GetListMarquesAsync();
            _cache.Set(ClefCacheMarques, marques, new MemoryCacheEntryOptions
            {
                AbsoluteExpirationRelativeToNow = TimeSpan.FromHours(1)
            });
        }

        return new SuggestTranscos(marque, marques, ListDimensionsVehicules.MARQUE);
    }

    public async Task<SuggestTranscos> CreerTranscoModeleAutomatique(string marque, string modele)
    {
        var clefCache = $"modeles-{marque}";

        if (_cache.TryGetValue(clefCache, out List<DatasCatalogueConstructeur> candidats) is false)
        {
            var candidatsNonTries = await
_catalogueConstructeurQueryHandler.GetListModelesByMarqueAsync(marque);
            candidats = candidatsNonTries
                .Select(c => new DatasCatalogueConstructeur { Libelle = c.ToUpper() })
                .ToList();

            _cache.Set(clefCache, candidats, new MemoryCacheEntryOptions
            {
                AbsoluteExpirationRelativeToNow = TimeSpan.FromHours(1)
            });
        }

        return new SuggestTranscos(modele, candidats, ListDimensionsVehicules.MODELE);
    }

    public async Task<SuggestTranscos> CreerTranscoSousModeleAutomatique(string marque, string modele)
    {
        var clefCache = $"sousmodele-{marque}";

        if (_cache.TryGetValue(clefCache, out List<DatasCatalogueConstructeur> candidats) is false)
        {
            var candidatsNonTries = await
_catalogueConstructeurQueryHandler.GetListSousModelesByMarqueAsync(marque);
            candidats = candidatsNonTries
                .Select(c => new DatasCatalogueConstructeur { Libelle = c.ToUpper()})
                .ToList();

            _cache.Set(clefCache, candidats, new MemoryCacheEntryOptions
            {
                AbsoluteExpirationRelativeToNow = TimeSpan.FromHours(1)
            });
        }

        return new SuggestTranscos(modele, candidats, ListDimensionsVehicules.SOUSMODELE);
    }
}
```

Annexe 5.3.10.c : Extrait de la classe SuggestTranscos

```
public class SuggestTranscos
{
    private readonly List<DatasCatalogueConstructeur> _datasRef;
    private readonly ListDimensionsVehicules _type;

    public SuggestTranscos(string valeurInitiale, List<DatasCatalogueConstructeur> datasRef,
        ListDimensionsVehicules type)
    {
        ValeurInitiale = valeurInitiale.ToUpper();
        _datasRef = datasRef;
        _type = type;
        Build();
    }

    public SuggestTranscos()
    {
    }

    public string ValeurInitiale { get; set; }
    public string ValeurRetenue { get; set; }
    public string SuggestionEgalite { get; set; }
    public string SuggestionContains { get; set; }
    public string SuggestionLevenshtein { get; set; }
    public string SuggestionSoundex { get; set; }
    public string SuggestionDoubleMetaphone { get; set; }
    public string LibelleSpecialChar { get; set; }
    public bool DoubleMetaphoneOk { get; private set; }
    public bool SoundexOk { get; private set; }
    public bool LevenshteinOk { get; private set; }
    public bool ContainsOk { get; private set; }
    public bool? SpecialCharOk { get; private set; }
    public bool? EgalityOk { get; private set; }
    public List<string> Suggestions { get; set; }
    public ListStatutsStrategieReconciliationModeleVehicule Statut { get; set; }

    public void Build()
    {
        Suggestions = new List<string>();
        CleaningEgality();
        CleaningSpecialChar();
        CleaningByContains();
        CleaningLevenshtein();
        CleaningSoundex();
        CleaningDoubleMetaphone();
        DetermineValeurRetenue();
        Statut = ListStatutsStrategieReconciliationModeleVehicule.INITIALISEE;
        Suggestions = Suggestions.Distinct().ToList();
    }

    private void DetermineValeurRetenue()
    {
        if ((bool)EgalityOk)
        {
            ValeurRetenue = SuggestionEgalite;
        }
        else
        {
            if (ContainsOk)
            {
                ValeurRetenue = SuggestionContains;
            }
            else
            {
                ValeurRetenue = "";
            }
        }
    }

    private void CleaningDoubleMetaphone()
    {
        var listResult = new List<DatasCatalogueConstructeur>();

        foreach (var dataRef in _datasRef)
        {
            if (DoubleMetaphoneMatch(dataRef.Libelle, LibelleSpecialChar))
            {
                listResult.Add(dataRef);
            }
        }

        if (listResult.Count == 1)
        {
            DoubleMetaphoneOk = true;
            SuggestionDoubleMetaphone = listResult[0].Libelle;
            Suggestions.Add(SuggestionDoubleMetaphone);
        }
        else if (listResult.Count > 1)
        {
            DoubleMetaphoneOk = false;
            SuggestionDoubleMetaphone = ConcatResults(listResult).Libelle;
            foreach (var libelle in listResult)
            {
                Suggestions.Add(libelle.Libelle);
            }
        }
        else
        {
            DoubleMetaphoneOk = false;
            SuggestionDoubleMetaphone = "";
        }
    }

    /// Ect....
}
```

Annexe 5.3.11.a : Le service d'enrichissement

```
public class EnrichissementInfosVehiculeService : IEnrichissementInfosVehiculeService
{
    private readonly ICatalogueConstructeurAdapter _handler;
    private readonly IReducer _reducer;

    public EnrichissementInfosVehiculeService(ICatalogueConstructeurAdapter handler)
    {
        _handler = handler;
        _reducer = new Reducer();
    }

    /// <summary>
    ///     Enrichir les informations du véhicule à partir du libellé
    ///     de la version si elles ne sont pas renseignées
    /// </summary>
    /// <param name="infosVehicule">Infos du véhicule à enrichir</param>
    /// <returns>Infos véhicules enrichies</returns>
    public async Task<ResultatEnrichissement> EnrichirInfosVehicule(IInfosVehicule infosVehicule, bool
versionNeedCompleted)
    {
        var dimensionsVehAEnrichir = _reducer.ReduceVehicule(infosVehicule, false);

        var process = new EnrichissementProcess(dimensionsVehAEnrichir, _handler);

        if (versionNeedCompleted)
        {
            var isVersionNotEmpty = process.CheckIfVersionNotEmpty();

            if (!isVersionNotEmpty)
            {
                return new ResultatEnrichissement
                {
                    ListDimensionsOriginalVehicule = dimensionsVehAEnrichir,
                    ListDimensionsEnrichies = null
                };
            }
        }

        process.DetermineDimensionsAEnrichir(versionNeedCompleted);

        var result = await process.ProcessEnrichissementAsync();

        return result;
    }
}
```

Annexe 5.3.11.b : Classe mère de l'enrichissement

```
public abstract class EnrichissementBase
{
    protected readonly string LibelleVersion;

    protected EnrichissementBase(string libelleVersion)
    {
        LibelleVersion = libelleVersion.ToUpper() + " ";
    }

    /// <summary>
    ///     Cherche dans le libellé si le pattern regex existe
    /// </summary>
    /// <param name="patterns"></param>
    /// <returns>La valeur correspondant à la regex</returns>
    public async Task<string> MatchingDimensionInLibelle(List<string> patterns)
    {
        foreach (var pattern in patterns)
        {
            var match = await Task.Run(() => Regex.Match(LibelleVersion, pattern));
            if (match.Success)
            {
                return match.Value;
            }
        }

        return "";
    }

    /// <summary>
    ///     Nettoie les infos de la dimensions pour garder les valeurs importantes
    /// </summary>
    /// <param name="wordsToReplace"></param>
    /// <param name="infoToClean"></param>
    /// <returns>String qui ne comprend plus d'informations superflues</returns>
    public static string CleanInFoDimension(List<string> wordsToReplace, string infoToClean)
    {
        foreach (var substr in wordsToReplace)
        {
            infoToClean = infoToClean.Replace(substr, "");
        }

        return infoToClean;
    }
}
```

Annexe 5.3.11.b : Classe gérant l'enrichissement du nombre de places

```
public class EnrichissementNbreDePlaces : EnrichissementBase, ITypeEnrichissement
{
    public EnrichissementNbreDePlaces(string libelleVersion) : base(libelleVersion)
    {
    }

    public async Task<IDimension?> EnrichirAsync()
    {
        var patterns = new List<string> { @"[1-9]\s?(PLACES|PLACE|PL)" };
        var wordsToReplace = new List<string> { "PLACES", " PLACES", "PLACE", " PLACE", "PL", " PL" };

        var nbrePlaces = await MatchingDimensionInLibelle(patterns);

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

            return new Dimension(int.Parse(nbrePlaces), ListDimensionsVehicules.NOMBRE_DE_PLACES);
        }

        return null;
    }
}
```

Annexe 5.3.12.a : Le service de réconciliation

```
public class ReconcilierVehiculeService : IReconcilierVehiculeService
{
    private readonly ICatalogueConstructeurAdapter _handler;

    public ReconcilierVehiculeService(ICatalogueConstructeurAdapter handler)
    {
        _handler = handler;
    }

    /// <summary>
    ///     Réconcilier infos véhicules avec id argus
    /// </summary>
    /// <param name="vehiculeATester">Infos du véhicule à tester</param>
    /// <param name="strategy">La stratégie réconciliation à utiliser</param>
    /// <returns>Reconciliation Modèle Véhicule</returns>
    public async Task<ReconciliationModeleVehicule> ReconcilierModeleVehicule(IInfosVehicule
vehiculeATester,
        IStrategieReconciliationModeleVehicule strategy)
    {
        var warnings = new List<Warning>();

        var reducer = new Reducer(strategy);

        var process = new ReconciliationProcess(vehiculeATester, strategy);

        var dimensionsVehATester = reducer.ReduceVehicule(vehiculeATester, true);

        var dataSetInitialHasOneResult = false;
        var listVehiculesCandidats = await _handler.GetListVehiculesCandidatsAsync(vehiculeATester,
strategy);

        if (listVehiculesCandidats.Count == 1)
        {
            dataSetInitialHasOneResult = true;
        }

        if (listVehiculesCandidats.Count == 0)
        {
            warnings.Add(
                process.GenerateWarning(
                "MARQUE,MODELE,DATE_MEC",
                "NOT_FOUND",
                "La requête initiale n'a pas renvoyé de résultat. Veui
modèle et la date MEC"
                ));
        }
        var dimensionsVehRef = reducer.ReduceListVehicules(listVehiculesCandidats, true);

        var dimensionsVehRefWithDistances = process.CalculDistances(dimensionsVehATester,
dimensionsVehRef);

        var dimensionsVehRefSansDistanceExtremes =
process.SupprimerDistanceSiValeurExtreme(dimensionsVehRefWithDistances);

        var dimensionsVehRefWithGoodEnergie = process.TriEnergie(dimensionsVehRefSansDistanceExtremes);

        if (dimensionsVehRefWithDistances.Count > 0 && dimensionsVehRefWithGoodEnergie.Count == 0)
        {
            warnings.Add(
                process.GenerateWarning(
                    "ENERGIE",
                    "NOT_FOUND",
                    "L'énergie en entrée ne correspond à aucun candidat ou est manquante."
                ));
        }

        var dimensionsAvecSelectionPrixCatalogue = process.SelectPrixCatalogue(dimensionsVehATester,
dimensionsVehRefWithGoodEnergie);

        var dimensionsVehRefSansDoublon =
process.SupprimerDoublonsIdArgus(dimensionsAvecSelectionPrixCatalogue);

        List<IListDimensionsVehicule> dimensionsVehRefWithSeuilDistanceMax;

        if (dimensionsAvecSelectionPrixCatalogue.Count != 0)
        {
            //var seuilDistanceMax = process.CalculateSeuilDistanceMax(dimensionsVehATester);
            dimensionsVehRefWithSeuilDistanceMax =
process.ApplySeuilDistanceMax(dimensionsVehRefSansDoublon);
        }
        else
        {
            dimensionsVehRefWithSeuilDistanceMax = dimensionsAvecSelectionPrixCatalogue;
        }

        var dimensionsVehRefTriee = process.TriParDistance(dimensionsVehRefWithSeuilDistanceMax);

        var dimensionsVehFinales = process.ApplyValeurMinimaleDeReprise(dimensionsVehRefTriee);

        return process.BuildResultatReconciliation(dimensionsVehFinales, dataSetInitialHasOneResult,
warnings);
    }
}
```

Annexe 5.3.12.b : Récupération des candidats dans le catalogue constructeur

```
public async Task<List<InfosVehiculePostgre>> GetListVehiculesCandidatsAsync(
    InfosVehiculePostgre vehiculeATester,
    int fenetreDateMec)
{
    var result = new List<InfosVehiculePostgre>();

    if (vehiculeATester.SousModele == null || vehiculeATester.SousModele == "")
    {
        if (vehiculeATester.DatePremiereMec != null)
        {
            var dateMec = DateTime.SpecifyKind((DateTime)vehiculeATester.DatePremiereMec,
DateTimeKind.Utc);

            var query =
                from version in dbContext.Versions
                join marque in dbContext.Marques on version.IdMarque equals marque.Id
                join modele in dbContext.Modeles on version.IdModele equals modele.Id
                join sousModele in dbContext.SousModeles on version.IdSousmodele equals sousModele.Id
                join energie in dbContext.Energies on version.IdEnergie equals energie.Id
                join prix in dbContext.Prix on version.Id equals prix.IdVersion
                join boiteVitesse in dbContext.BoiteVitesses on version.IdBoitevitesse equals
boiteVitesse.Id
                join typeBoiteVitesse in dbContext.TypeBoiteVitesses on boiteVitesse.IdTypeboitevitesse
equals
                    typeBoiteVitesse.Id
                join finition in dbContext.Finitions on version.IdFinition equals finition.Id
                join generation in dbContext.Generations on version.IdGeneration equals generation.Id
                join phase in dbContext.Phases on version.IdPhase equals phase.Id
                where marque.Libelle == vehiculeATester.Marque
                where modele.Libelle.ToUpper() == vehiculeATester.Modele
                where version.Datedebut <= DateOnly.FromDateTime(dateMec.AddMonths(6)) &&
                      version.Datefin >= DateOnly.FromDateTime(dateMec.AddMonths(-fenetreDateMec)) ||
                      version.Datedebut <= DateOnly.FromDateTime(dateMec.AddMonths(6)) &&
                      version.Datefin == null
                select new InfosVehiculePostgre
                {
                    IdArgus = (int?)version.Idexterne,
                    Marque = marque.Libelle,
                    Modele = modele.Libelle,
                    SousModele = sousModele.Libelle,
                    DatePremiereMec = dateMec,
                    Energie = energie.Libelle,
                    BoiteVitesse = boiteVitesse.Libelle,
                    TypeBoiteVitesse = typeBoiteVitesse.Libelle,
                    PuissanceFiscale = version.Nombrechevauxfiscaux,
                    PuissanceDin = version.Puissancedin,
                    NombreDePortes = version.Nombreportes,
                    NombreDePlaces = version.Habitabilitenombreplaces,
                    EmissionCo2 = (float?)version.Consommationemissionco2,
                    Cylindree = version.Moteurcylindreecm3,
                    Carrosserie = version.Libellecarrosseriereferentiel,
                    Generation = generation.Libelle,
                    Phase = phase.Libelle,
                    Version = version.Libellelong,
                    PrixCatalogue = (float?)prix.Prixttc,
                    DateDebutPrixCatalogue =
prix.Datedebut.GetValueOrDefault().ToDateTime(TimeOnly.Parse("00:00PM")),
                    DateFinPrixCatalogue =
prix.Datefin.GetValueOrDefault().ToDateTime(TimeOnly.Parse("00:00PM")),
                    Poids = version.Poidsavide,
                    Consommation = (float?)version.Consommationmixte,
                    Finition = finition.Libelle,
                    DateDebut =
version.Datedebut.GetValueOrDefault().ToDateTime(TimeOnly.Parse("00:00PM")),
                    DateFin = version.Datefin.GetValueOrDefault().ToDateTime(TimeOnly.Parse("00:00PM"))
                };

            result = await query.ToListAsync();
        }
    }
```

Annexe 5.3.12.c : Suppression des distances avec valeurs extrêmes

```
 public List<IListDimensionsVehicule> SupprimerDistanceSiValeurExtreme(List<IListDimensionsVehicule>
dimensionsVehRefWithDistances)
 {
     var distanceFinitionTropHaute = false;
     var listDimensionFinition = new List<IDimension>();

     foreach (var listDimension in dimensionsVehRefWithDistances)
     {
         var finitionDimension = listDimension.Dimensions.FirstOrDefault(d => d.Type ==
CommonLists.ListDimensionsVehicules.FINITION);

         if (finitionDimension != null)
         {
             listDimensionFinition.Add(finitionDimension);
         }
     }

     distanceFinitionTropHaute = listDimensionFinition.Any() && listDimensionFinition.All(dimension =>
dimension.DistanceDimension >= 0.6);

     if(!distanceFinitionTropHaute)
     {
         return dimensionsVehRefWithDistances;
     }

     var listDimensionsCorrigees = new List<IListDimensionsVehicule>();

     foreach (var listDimension in dimensionsVehRefWithDistances)
     {
         var dimensionFinition = listDimension.Dimensions.Where(d => d.Type ==
CommonLists.ListDimensionsVehicules.FINITION).FirstOrDefault();

         if (dimensionFinition != null)
         {
             var distance = dimensionFinition.DistanceDimension;
             var ponderation = dimensionFinition.Ponderation;

             if (distance != null)
             {
                 listDimension.Distance -= ((float)distance * ponderation);
                 listDimension.Dimensions.Where(d => d.Type ==
CommonLists.ListDimensionsVehicules.FINITION).FirstOrDefault().DistanceDimension = null;
             }
         }

         listDimensionsCorrigees.Add(listDimension);
     }

     return listDimensionsCorrigees;
 }
```

Annexe 5.3.12.d : Tri de l'énergie

```
    public List<IListDimensionsVehicule> TriEnergie(List<IListDimensionsVehicule>
dimensionsVehRefWithDistances)
    {
        return dimensionsVehRefWithDistances
            .Where(v => v.Dimensions.Any(d => d.Type == CommonLists.ListDimensionsVehicules.ENERGIE &&
d.DistanceDimension == 0F))
            .ToList();
    }
```

Annexe 5.3.12.e : Sélection du prix catalogue

```
    public List<IListDimensionsVehicule> SelectPrixCatalogue(IListDimensionsVehicule
dimensionsVehATester, List<IListDimensionsVehicule> dimensionsVehRefWithDistances)
    {
        var dateMec = dimensionsVehATester?.InfosVehicule?.DatePremiereMec;

        if (dateMec != null)
        {
            foreach (var group in dimensionsVehRefWithDistances.GroupBy(d => d.InfosVehicule.IdArgus))
            {
                var prixWithDateInRange = group
                    .Where(veh => veh.InfosVehicule.DateDebutPrixCatalogue <= dateMec && dateMec <=
veh.InfosVehicule.DateFinPrixCatalogue)
                    .OrderBy(veh => (veh.InfosVehicule.DateDebutPrixCatalogue - dateMec)?.Duration())
                    .ThenBy(veh => (veh.InfosVehicule.DateFinPrixCatalogue - dateMec)?.Duration())
                    .FirstOrDefault();

                if (prixWithDateInRange != null)
                {
                    foreach(var veh in group)
                    {
                        veh.InfosVehicule.PrixCatalogue =
prixWithDateInRange.InfosVehicule.PrixCatalogue;
                    }
                }
                else
                {
                    var prixWithNearestDate = group
                        .OrderBy(veh => (veh.InfosVehicule.DateDebutPrixCatalogue -
dateMec)?.Duration())
                        .ThenBy(veh => (veh.InfosVehicule.DateFinPrixCatalogue - dateMec)?.Duration())
                        .FirstOrDefault();

                    if (prixWithNearestDate != null)
                    {
                        foreach(var veh in group)
                        {
                            veh.InfosVehicule.PrixCatalogue =
prixWithNearestDate.InfosVehicule.PrixCatalogue;
                        }
                    }
                }
            }
        }

        return dimensionsVehRefWithDistances;
    }
```

Annexe 5.3.12.f : Suppression des doublons d'id argus

```
    public List<IListDimensionsVehicule> SupprimerDoublonsIdArgus(List<IListDimensionsVehicule>
dimensionsVehRefTriee)
    {
        return dimensionsVehRefTriee
            .GroupBy(v => v.InfosVehicule.IdArgus)
            .Select(g => g.OrderBy(item => item.Distance).FirstOrDefault())
            .ToList();
    }
```

Annexe 5.3.12.g : Application du seuil de distance maximale

```
public List<IListDimensionsVehicule> ApplySeuilDistanceMax(
    List<IListDimensionsVehicule> dimensionsVehRefSansDoublon)
{
    var distanceMin = dimensionsVehRefSansDoublon.Select(d => d.Distance).Min();
    var distanceTri = distanceMin + (float)_strategy.SeuilDistanceMaximale * distanceMin / 100;
    return dimensionsVehRefSansDoublon
        .Where(d => d.Distance <= distanceTri)
        .ToList();
}
```

Annexe 5.3.12.h : Tri par distance

```
    public List<IListDimensionsVehicule> TriParDistance(List<IListDimensionsVehicule>
dimensionsVehRefWithDistances)
    {
        return dimensionsVehRefWithDistances
            .OrderBy(d => d.Distance)
            .ToList();
    }
```

Annexe 5.3.12.i : Application de la valeur minimale de reprise

```
    public List<IListDimensionsVehicule> ApplyValeurMinimaleDeReprise(List<IListDimensionsVehicule>
dimensionsVehRefTriee)
    {
        if (!_strategy.ForcageValeurMinimaleDeReprise)
        {
            return dimensionsVehRefTriee;
        }
        else
        {
            var result = new List<IListDimensionsVehicule>();

            var prixMinVehicule = dimensionsVehRefTriee?
                .Where(v => v != null && v.Dimensions != null && v.Dimensions.Any(d => d != null &&
d.Type == CommonLists.ListDimensionsVehicules.PRIX_CATALOGUE && d.Value != null))
                .OrderBy(v => Convert.ToDecimal(v.Dimensions.First(d => d.Type ==
CommonLists.ListDimensionsVehicules.PRIX_CATALOGUE).Value))
                .FirstOrDefault();

            if (prixMinVehicule != null)
            {
                result.Add(prixMinVehicule);
            }

            return result;

        }
    }
```

Annexe 5.3.13.a : Moteur du calcul de distance

```
    public List<IListDimensionsVehicule> MatchingDimensionsAndCalculDistance()
    {
        foreach (var vehDimensionRef in _listDimensionsVehCandidats)
            foreach (var dimRef in vehDimensionRef.Dimensions)
            {
                if (_dimensionsTestedVeh != null && _dimensionsTestedVeh.Dimensions != null)
                {
                    var dimTest = _dimensionsTestedVeh.Dimensions.FirstOrDefault(dimTest => dimRef.Type
== dimTest.Type);
                    if (dimTest != null)
                    {
                        dimRef.CalculerDistance(dimTest);
                        vehDimensionRef.CalculDistanceGenerale(dimRef);
                    }
                }

            }

        return _listDimensionsVehCandidats;
    }
```

Annexe 5.3.13.b : La Calcul Distance Factory

```
public class CalculDistanceFactory
{
    private readonly IDimension _dimRef;
    private readonly IDimension _dimTest;

    public CalculDistanceFactory(IDimension dimTest, IDimension dimRef)
    {
        _dimTest = dimTest;
        _dimRef = dimRef;
    }

    public ICalculDistanceType? Build()
    {
        switch (_dimTest.Type)
        {
            case ListDimensionsVehicules.MARQUE:
            case ListDimensionsVehicules.MODELE:
            case ListDimensionsVehicules.CARROSSERIE:
            case ListDimensionsVehicules.FINITION:
            case ListDimensionsVehicules.BOITE_DE_VITESSE:
            case ListDimensionsVehicules.GENERATION:
            case ListDimensionsVehicules.PHASE:
            case ListDimensionsVehicules.SOUSMODELE:
            case ListDimensionsVehicules.TYPE_BOITE_DE_VITESSE:
                return new CalculDistanceString(_dimRef, _dimTest);

            case ListDimensionsVehicules.VERSION:
                return new CalculDistanceVersion(_dimRef, _dimTest);

            case ListDimensionsVehicules.NOMBRE_DE_PLACES:
            case ListDimensionsVehicules.NOMBRE_DE_PORTES:
                return new CalculDistanceInteger(_dimRef, _dimTest);

            case ListDimensionsVehicules.CYLINDREE:
            case ListDimensionsVehicules.CONSOMMATION:
            case ListDimensionsVehicules.EMISSION_CO2:
            case ListDimensionsVehicules.POIDS:
            case ListDimensionsVehicules.PUISSANCE_DIN:
            case ListDimensionsVehicules.PUISSANCE_FISCALE:
            case ListDimensionsVehicules.PRIX_CATALOGUE:
                return new CalculDistanceFloat(_dimRef, _dimTest);

            case ListDimensionsVehicules.ENERGIE:
                return new CalculDistanceEnergie(_dimRef, _dimTest);

            default:
                return null;
        }
    }
}
```

Annexe 5.3.13.c : La Classe mère du calcul de distances

```
public abstract class CalculDistanceBase
{
    private readonly IDimension _dimRef;
    private readonly IDimension _dimTest;

    protected CalculDistanceBase(IDimension dimTest, IDimension dimRef)
    {
        _dimTest = dimTest;
        _dimRef = dimRef;
    }

    public virtual float CalcDistance()
    {
        return _dimTest.Value.Equals(_dimRef.Value) ? 0 : 1;
    }
}
```

Annexe 5.3.13.d : Le calcul des distances entre numériques flottants

```
public class CalculDistanceFloat : CalculDistanceBase, ICalculDistanceType
{
    private readonly float _valueRef;
    private readonly float _valueTest;

    public CalculDistanceFloat(IDimension dimRef, IDimension dimTest) : base(dimTest, dimRef)
    {
        _valueRef = (float)dimRef.Value;
        _valueTest = (float)dimTest.Value;
    }

    public override float CalcDistance()
    {
        var distance = base.CalcDistance();
        if (Math.Abs(distance - 1F) < 0.00001F)
        {
            distance = PercentDifferenceDimension();
        }

        return distance;
    }

    /// <summary>
    ///     Calcul le pourcentage de différence entre les deux nombres. Nécessairement entre 0 et 1
représentant la distance
    /// </summary>
    /// <returns>Float entre 0 et 1</returns>
    private float PercentDifferenceDimension()
    {
        var diff = Math.Abs(_valueRef - _valueTest);

        return _valueRef > _valueTest ? diff / _valueRef : diff / _valueTest;
    }
}
```

Annexe 5.3.13.e : Le calcul des distances entre chaines de caractères

```
public class CalculDistanceString : CalculDistanceBase, ICalculDistanceType
{
    private readonly string _valueRef;
    private readonly string _valueTest;

    public CalculDistanceString(IDimension dimRef, IDimension dimTest) : base(dimTest, dimRef)
    {
        _valueRef = (string)dimRef.Value;
        _valueTest = (string)dimTest.Value;
    }

    public override float CalcDistance()
    {
        var distance = base.CalcDistance();

        if (Math.Abs(distance - 1F) < 0.00001F)
        {
            distance = Levenshtein.GetLevenshteinIndice(_valueTest, _valueRef);
        }

        return distance;
    }
}
```

Annexe 5.3.13.f : Le calcul de la distance de Levenshtein normalisée

```
public static class Levenshtein
{
    public static float GetLevenshteinIndice(string value1, string value2)
    {
        var levenshteinDistance = Fastenshtein.Levenshtein.Distance(value1, value2);

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

    public static int GetLevenshteinDistance(string value1, string value2)
    {
        if (value1 != null && value2 != null)
        {
            return Fastenshtein.Levenshtein.Distance(value1, value2);
        }

        return 10000;
    }
}
```

Annexe 5.3.13.g : Le calcul de la distance pour l'énergie

```
public override float CalcDistance()
{
    if(_valueTest.Contains("HYBRIDE RECHARGEABLE") && _valueRef.Contains("HYBRIDE RECHARGEABLE"))
    {
        return 0F;
    }
    if (_valueTest.Contains("HYBRIDE") && _valueRef.Contains("HYBRIDE"))
    {
        if (!_valueTest.Contains("rechargeable") && !_valueTest.Contains("rechargeable"))
        {
            return 0F;
        }
    }

    if (_valueTest.Contains("ELECTRIQUE") && _valueRef.Contains("ELECTRIQUE"))
    {
        if (!_valueRef.Contains("HYBRIDE"))
        {
            return 0F;
        }
    }

    if (_valueTest.Contains("ESSENCE") && _valueRef.Contains("ESSENCE"))
    {
        return 0F;
    }
    else if (_valueTest.Contains("DIESEL") && _valueRef.Contains("DIESEL"))
    {
        return 0F;
    }

    var distance = base.CalcDistance();

    if (Math.Abs(distance - 1F) < 0.00001F)
    {
        distance = Levenshtein.GetLevenshteinIndice(_valueTest, _valueRef);
    }

    return distance;
}
```

Annexe 5.3.13.h : Le calcul de la distance pour le champ Version

```
public override float CalcDistance()
{
    var distance = base.CalcDistance();
    if (Math.Abs(distance - 1F) < 0.00001F)
    {
        distance = CalcDistanceCommonWords();
    }

    return distance;
}

/// <summary>
///     Calcul la distance basée sur le nombre de mots en commun dans les libellés
/// </summary>
/// <returns>Distance "mots en commun"</returns>
private float CalcDistanceCommonWords()
{
    var calcDistanceByWords = new CalculDistanceLibelle(_valueTest, _valueRef);
    calcDistanceByWords.Execute();
    var distance = calcDistanceByWords.GetDistance();
    return distance;
}
```

Annexe 5.3.13.i : Classe utilitaire pour les opérations sur les chaines de caractères

```
public static class CompareStrings
{
    /// <summary>
    ///     Découpe la chaine de caractères en liste de mots
    /// </summary>
    /// <param name="libelleVehicule"></param>
    /// <returns></returns>
    public static List<string> CutString(string libelleVehicule)
    {
        var resultTab = new List<string>();

        if (libelleVehicule != "")
        {
            resultTab = libelleVehicule.Split(" ").ToList();
        }

        return resultTab;
    }

    /// <summary>
    ///     Renvoie le nombre de mots communs dans les deux listes de mots
    /// </summary>
    /// <param name="listWordVehicule1">Liste de mots 1</param>
    /// <param name="listWordVehicule2">Liste de mots 2</param>
    /// <returns>Nombre de mots communs</returns>
    public static float GetSumDistances(List<string> listWordVehicule1, List<string> listWordVehicule2)
    {
        return listWordVehicule1.Where(item => listWordVehicule2
                .Any(word => word == item))
            .Aggregate(0, (current, item) => current + 1);
    }
}
```

Annexe 5.3.13.j : Calcul de la distance de Jaccard normalisée

```
public class CalculDistanceLibelle
{
    private readonly string _libelleVehicule;
    private readonly string _libelleVehReferentiel;
    private List<string> _listWordReferentiel;
    private List<string> _listWordVehicule;
    private float _sumDistances;

    public CalculDistanceLibelle(string libelleVehicule, string libelleVehReferentiel)
    {
        _libelleVehicule = libelleVehicule;
        _libelleVehReferentiel = libelleVehReferentiel;
        _listWordReferentiel = new List<string>();
        _listWordVehicule = new List<string>();
    }

    /// <summary>
    ///     Lance le calcul de la distance de la version
    /// </summary>
    public void Execute()
    {
        _listWordVehicule = CompareStrings.CutString(_libelleVehicule);
        _listWordReferentiel = CompareStrings.CutString(_libelleVehReferentiel);

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

    /// <summary>
    ///     Renvoie la proportion de mot en commun dans les deux listes de mots
    /// </summary>
    /// <returns>Proportion (entre 0 et 1)</returns>
    public float GetDistance()
    {
        if (_listWordReferentiel.Count != 0 && _listWordVehicule.Count != 0)
        {
            var divide = _listWordReferentiel.Count >= _listWordVehicule.Count
                ? _listWordReferentiel.Count
                : _listWordVehicule.Count;
            return 1 - _sumDistances / divide;
        }

        return 1F;
    }
}
```

Annexe 5.3.14.a : Appel au service de récupération des données véhicules par immatriculation

```
public class RestSharpAccesAntsRequests : IAccesAntsRequests
{
    public RestSharpAccesAntsRequests(string accesAntsUrl)
    {
        Client = new RestClient(accesAntsUrl);
    }

    public RestClient Client { get; set; }
    public RestRequest Request { get; set; }

    public void CreateRequest(string user, string service, string plaque)
    {
        Request = new RestRequest("recupererinformationvehicule", Method.Post)
        {
            RequestFormat = DataFormat.Json
        };

        var headers = new Dictionary<string, string>
        {
            { "Content-Type", "application/json" },
            { "x-utilisateur", user },
            { "x-serviceInitial", service }
        };

        Request.AddHeaders(headers);

        var data = new RecupererInformationVehiculeAntsCommand
        {
            Plaque = plaque
        };

        Request.AddJsonBody(data);
    }


    public async Task<(RecupererInformationVehiculeAntsCommandResult, string)> LaunchRequest()
    {
        try
        {
            var response = await Client.PostAsync(Request);

            if (response.IsSuccessful)
            {
                var result = response.Content;
                var jobject = JObject.Parse(result);
                var datas =
                JsonConvert.DeserializeObject<RecupererInformationVehiculeAntsCommandResult>(
                    jobject["Data"].ToString(), new StringEnumConverter());

                return (datas, response.StatusCode.ToString());
            }

            return (null, response.StatusCode.ToString());
        }
        catch (Exception ex)
        {
            return (null, ex.Message);
        }
    }
}
```

Annexe 5.3.14.b : Appel au service externe SIV Argus

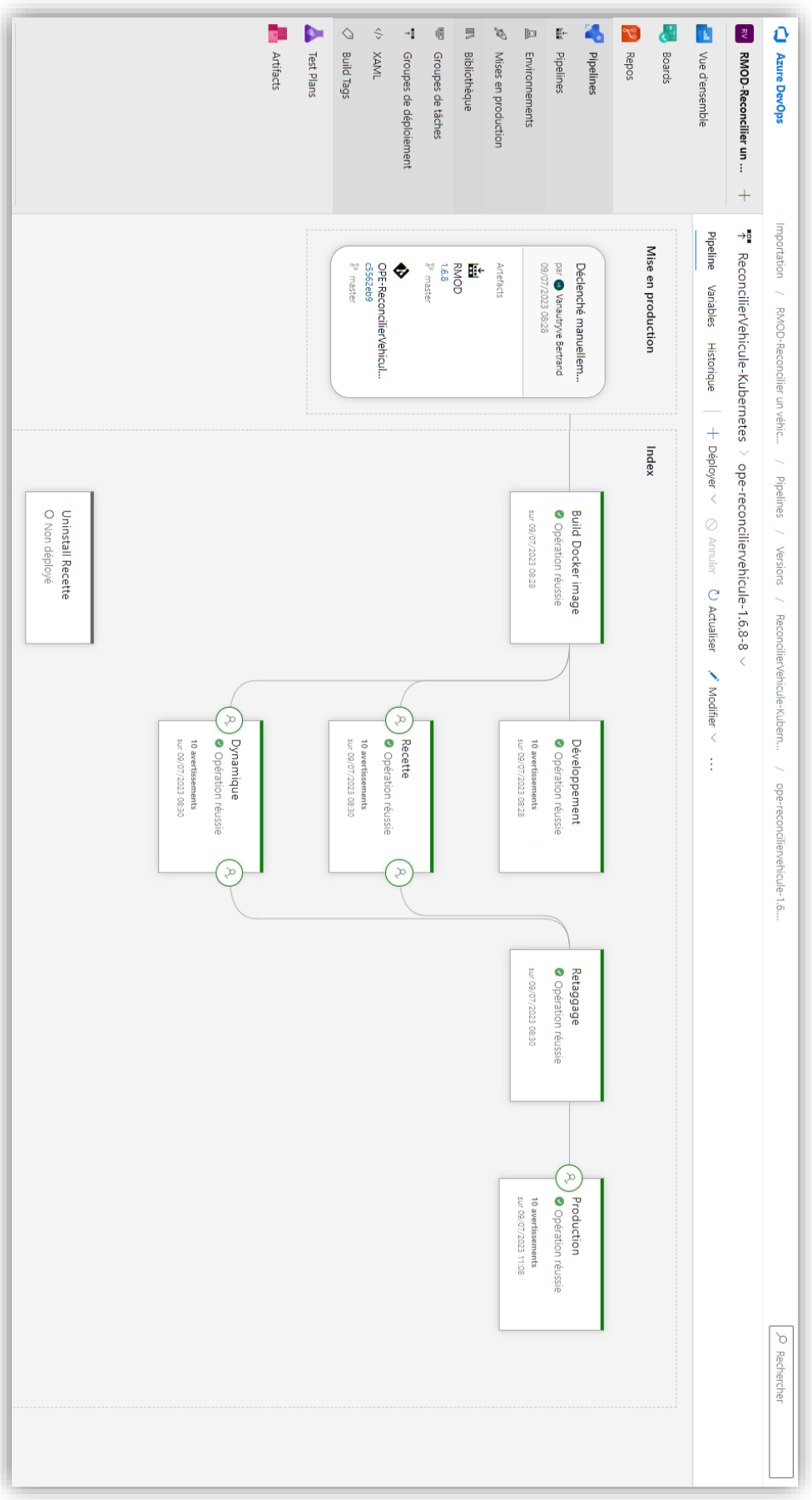
```
public async Task<OneOf<ArgusErreur, ArgusPlaqueNonTrouvee, ResultatMatchingArgus>>
    RechercherParImmatriculation(
        Immatriculation immatriculation,
        CancellationToken cancellationToken)
{
    // 1er appel: récupération d'un bearer d'authentification
    var resultatGenBearer = await RecupererTokenAsync(cancellationToken);

    if (resultatGenBearer.TryPickT1(out var _, out var bearer))
    {
        return new ArgusErreur
        {
            Erreur = "erreur à la récupération du bearer argus"
        };
    }

    // 2nd appel: récupération de l'identifiant de matching
    var resultatMatching = await RecupererDonneesMatchingArgusAsync(
        immatriculation.Libelle,
        bearer,
        cancellationToken);

    if (resultatMatching.TryPickT1(out _, out var matching))
    {
        return new ArgusErreur
        {
            Erreur = $"erreur au matching du véhicule '{immatriculation.Libelle}'"
        };
    }

    // 3ème appel: récupération des ressources (CG / candidats) liés au matching
    var resultatRecupCandidats = await RecupererCaracteristiquesAsync(
        matching.IdentifiantMatching.Id,
        bearer,
        cancellationToken);

    if (resultatRecupCandidats.TryPickT0(
            out var caracteristiquesMatchingArgus,
            out var erreursRecuperationCaracs) is false)
    {
        if (erreursRecuperationCaracs.TryPickT0(out var plaqueNonTrouvee, out _))
        {
            return plaqueNonTrouvee;
        }

        return new ArgusErreur();
    }

    return caracteristiquesMatchingArgus;
}
```

Annexe 5.4.2 : Exemple de test d'intégration

```
public class ReconcilierModeleImmatriculationTests : IClassFixture<WebApplicationFactory<Startup>>
{
    private readonly WebApplicationFactory<Startup> _factory;

    public ReconcilierModeleImmatriculationTests(WebApplicationFactory<Startup> factory)
    {
        _factory = factory;
    }

    [Theory]
    [InlineData("AZEAEZAEA")]
    [InlineData("")]
    public async Task RequetesAvecMauvaisFormatPlaques_Retournent400(string plaque)
    {
        // Arrange
        var requete = new HttpRequestMessage(
            HttpMethod.Post,
            "/reconcilierModeleImmatriculation");

        requete.Headers.Add("x-utilisateur", "tester");
        requete.Headers.Add("x-serviceInitial", "testApp");


        var content = JsonContent.Create(new { plaque });
        requete.Content = content;

        var client = _factory.CreateClient();

        // Act
        var response = await client.SendAsync(requete);

        // Assert
        response.StatusCode.Should().Be(HttpStatusCode.BadRequest);
    }
```

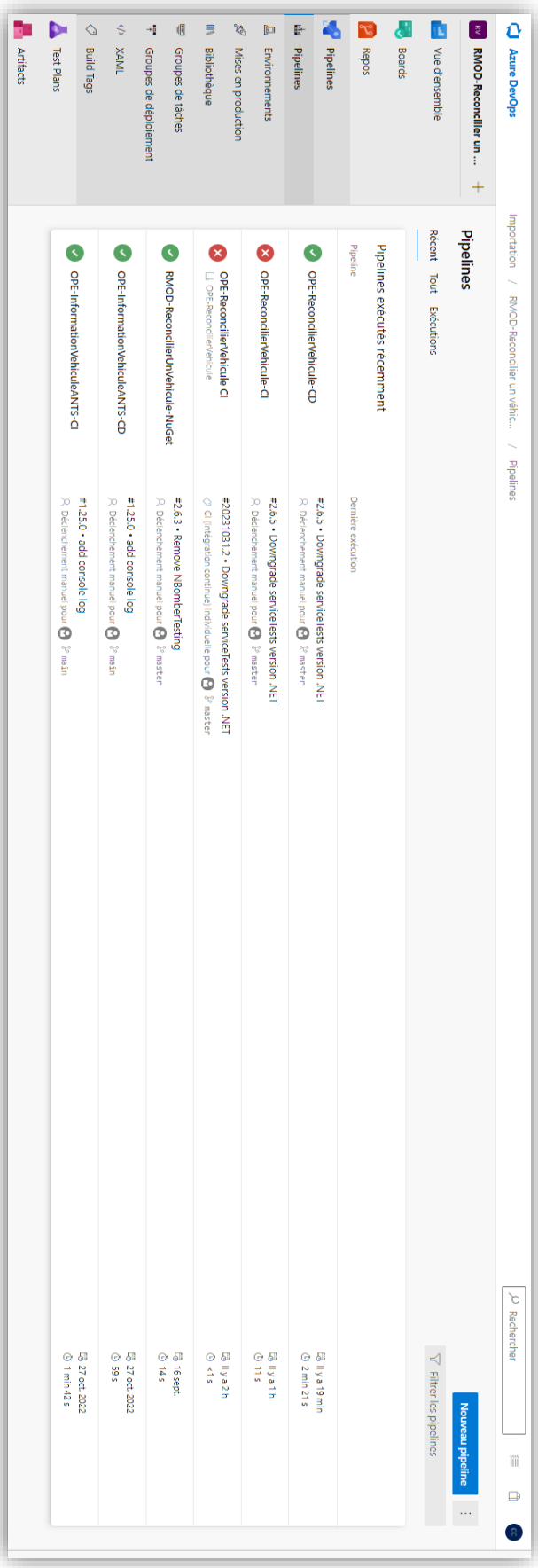
```
    [Theory]
    [InlineData("CG-514-AD")]
    [InlineData("CG514AD")]
    [InlineData("721CQX59")]
    [InlineData("1871ZH56")]
    public async Task RequetesAppelantSivArgusPlaques_Retournent200(string plaque)
    {
        // Arrange
        var mockSivArgus = new Mock<ISivArgusService>();

        mockSivArgus
            .Setup(argusService =>
                argusService.RechercherParImmatriculation(It.Is<Immatriculation>(i => i.Libelle ==
                                                                                      plaque),
                    It.IsAny<CancellationToken>()))
            .Returns(Task.FromResult(
                OneOf<ArgusErreur, ArgusPlaqueNonTrouvee, ResultatMatchingArgus>.FromT2(
                    new ResultatMatchingArgus
                    {
                        Ressources = new List<Ressource>
                        {
                            new()
                            {
                                TypeRessource = "registration-cards", Caracteristiques = new
                                                                                         Caracteristiques
                                {
                                    Couleur = "GRIS C"
                                }
                            },
                            new()
                            {
                                TypeRessource = "candidates", Caracteristiques = new
                                                                                 Caracteristiques
                                {
                                    IdArgus = 972356
                                }
                            }
                        }
                    })));

        var factory = _factory.WithWebHostBuilder(builder =>
        {
            builder.ConfigureServices(svc => { svc.AddSingleton(mockSivArgus.Object); });
        });


        var requete = new HttpRequestMessage(
            HttpMethod.Post,
            "/reconcilierModeleImmatriculation");


        requete.Headers.Add("x-utilisateur", "tester");
        requete.Headers.Add("x-serviceInitial", "testApp");


        var content = JsonContent.Create(new { immatriculation = plaque });
        requete.Content = content;

        var client = factory.CreateClient();

        // Act
        var response = await client.SendAsync(requete);

        // Assert
        response.StatusCode.Should().Be(HttpStatusCode.OK);
    }
}
```

Annexe 5.5.1 : Les différents environnements sur Azure DevOps

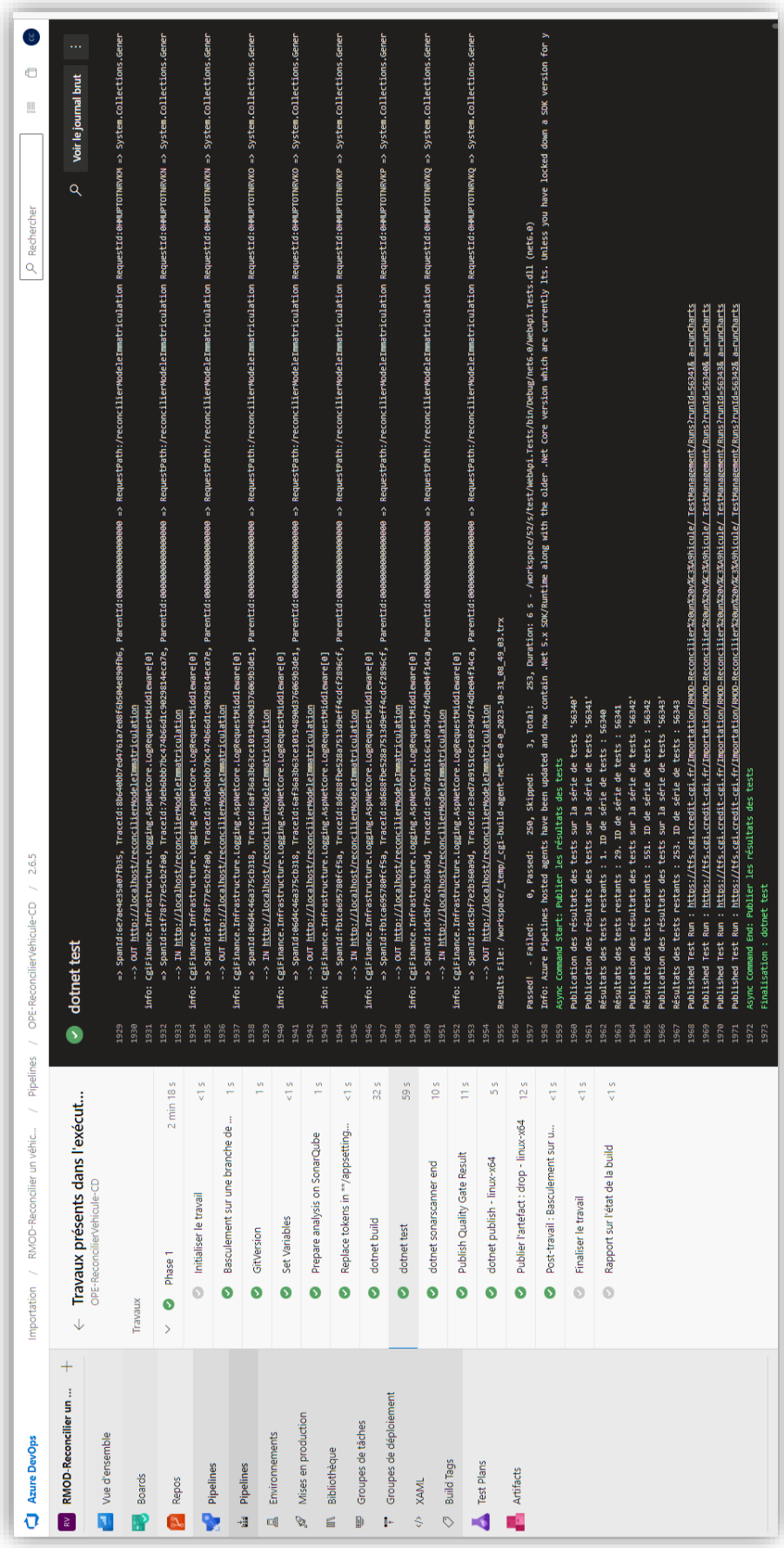


Annexe 5.5.4.a : Les étiquettes de version de l'application

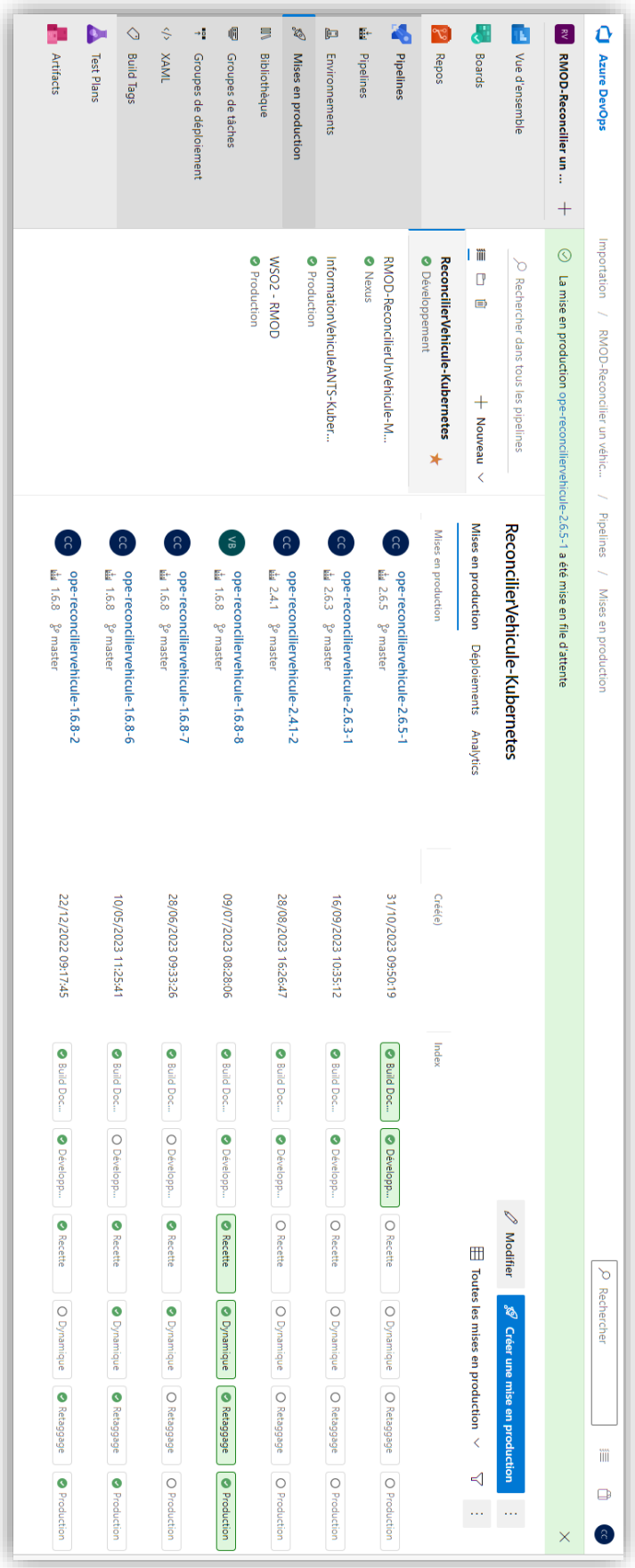
Annexe 5.5.4.b : Le pipeline de build de l'application



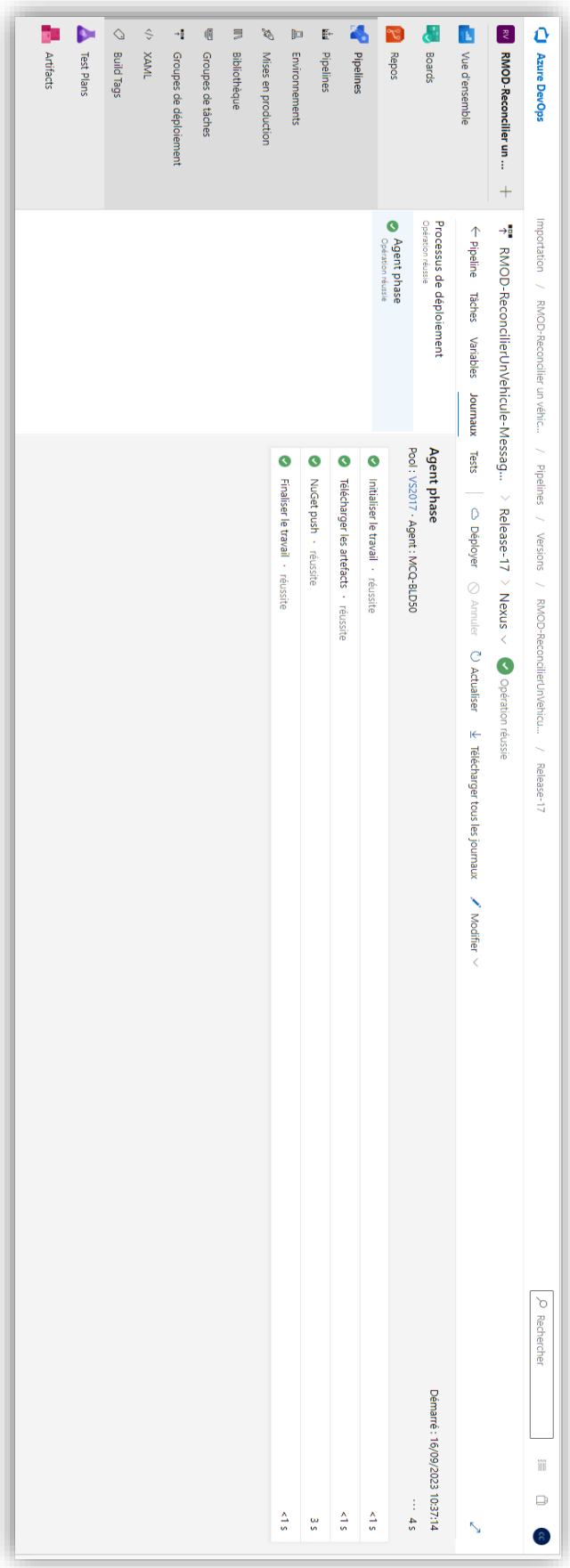
Annexe 5.5.4.c : Le détail du pipeline de build de l'application



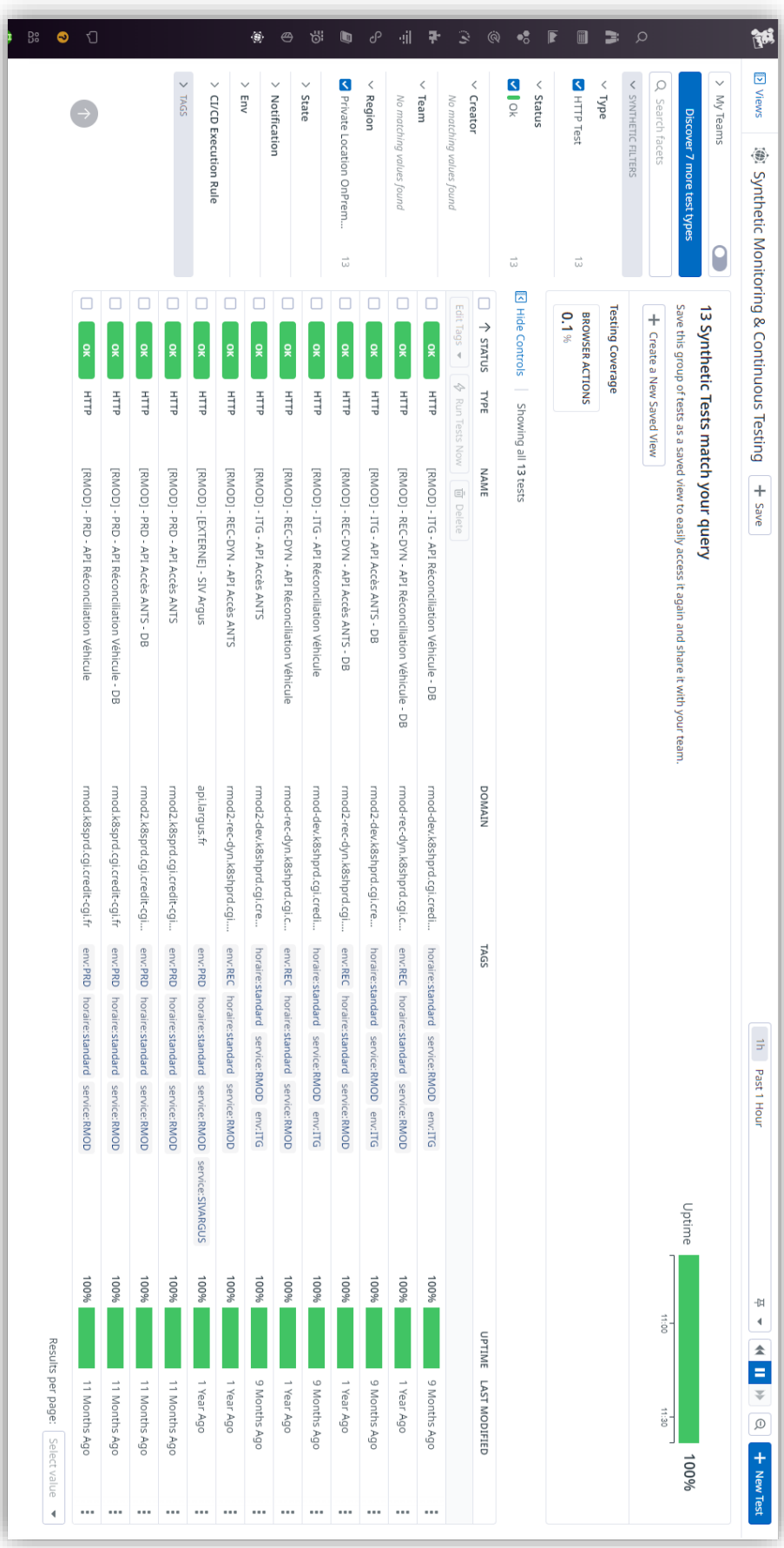
Annexe 5.5.4.d : Le pipeline de déploiement de l'application



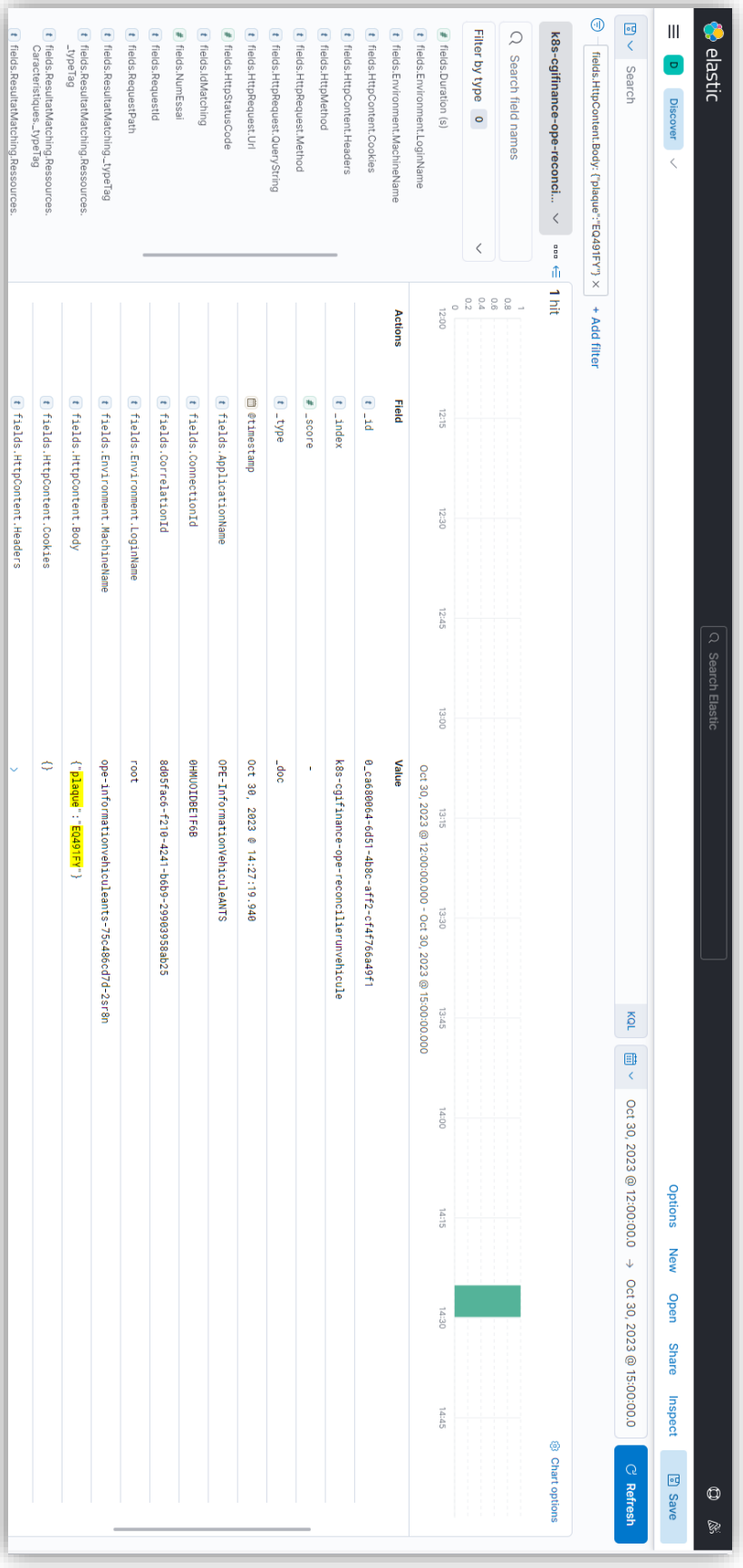
Annexe 5.5.4.e : Le pipeline de déploiement du Nuget Message



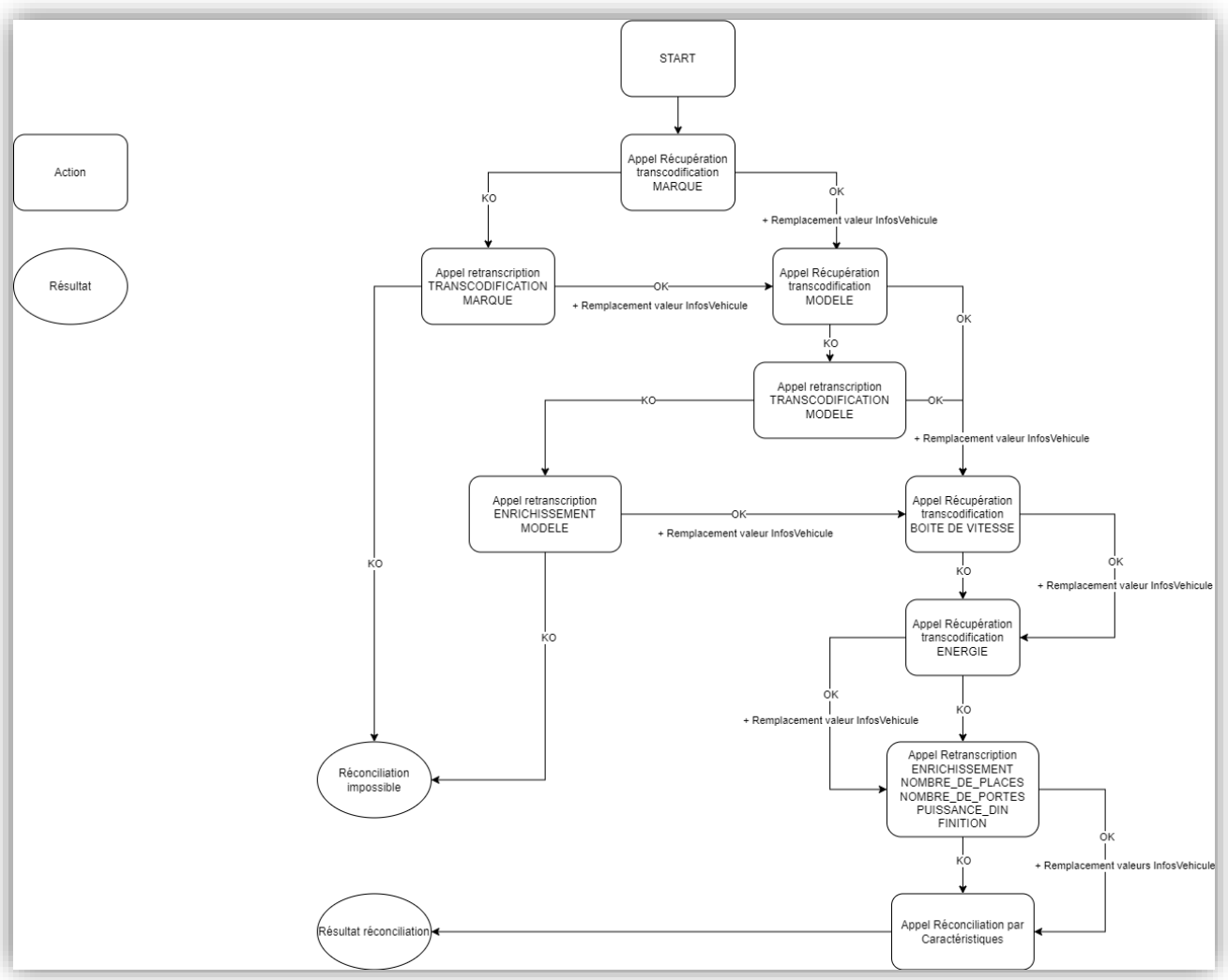
Annexe 6.1.3 : Liste des tests des healthchecks dans Datadog



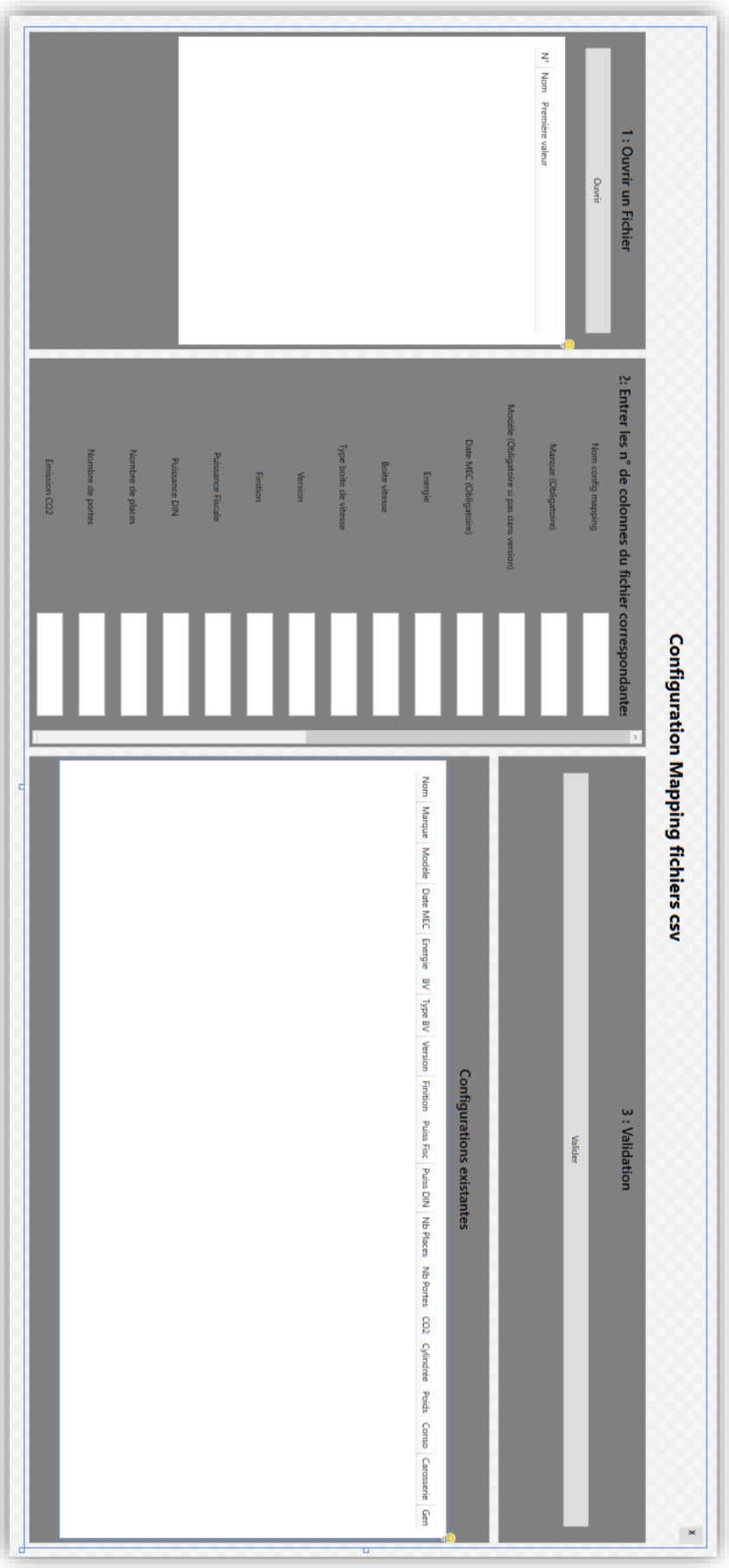
Annexe 6.3.2 : Interface ELK pour la gestion des logs



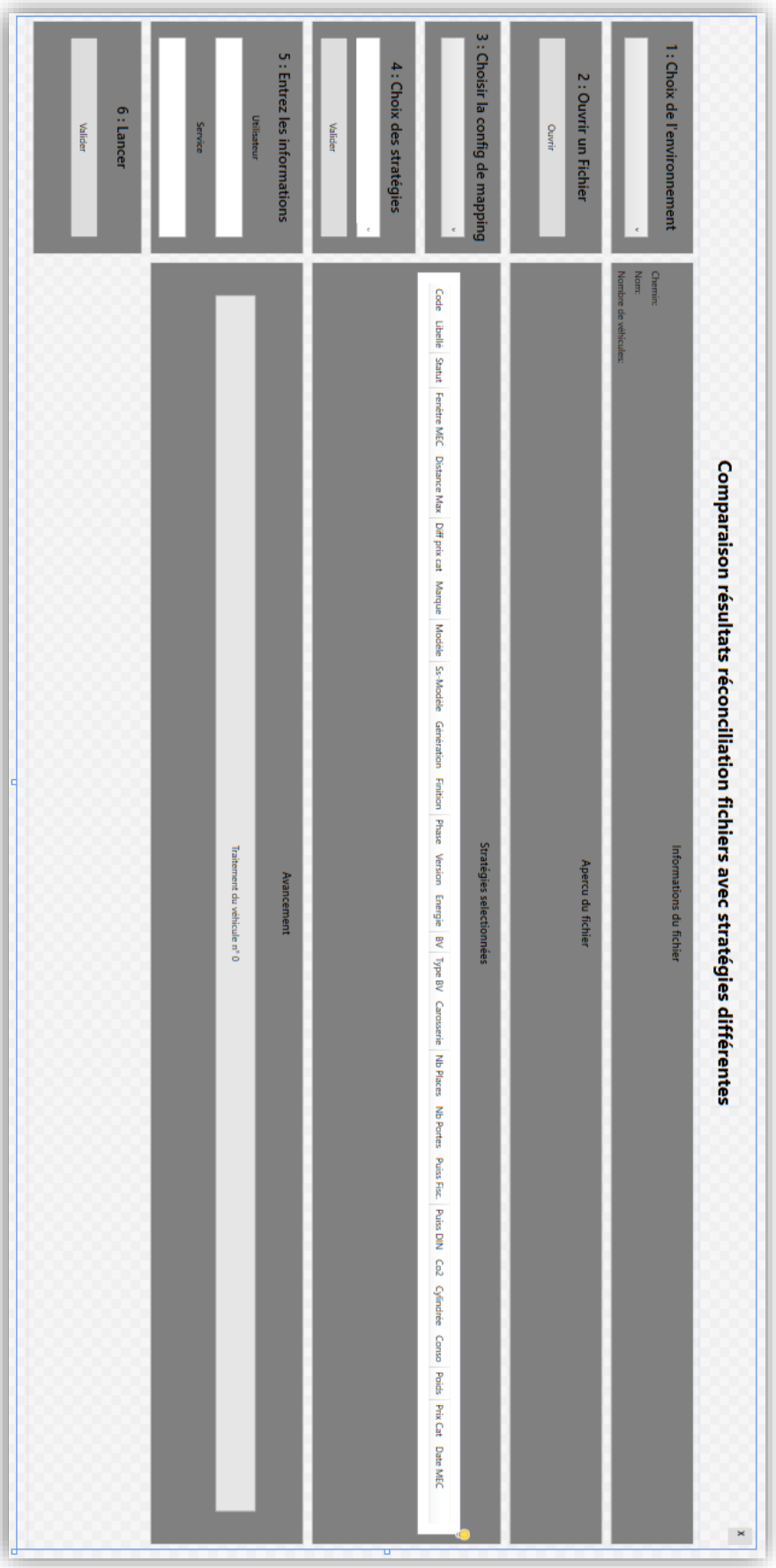
Annexe 6.5 : Arbre de décision pour l'utilisation de RMOD



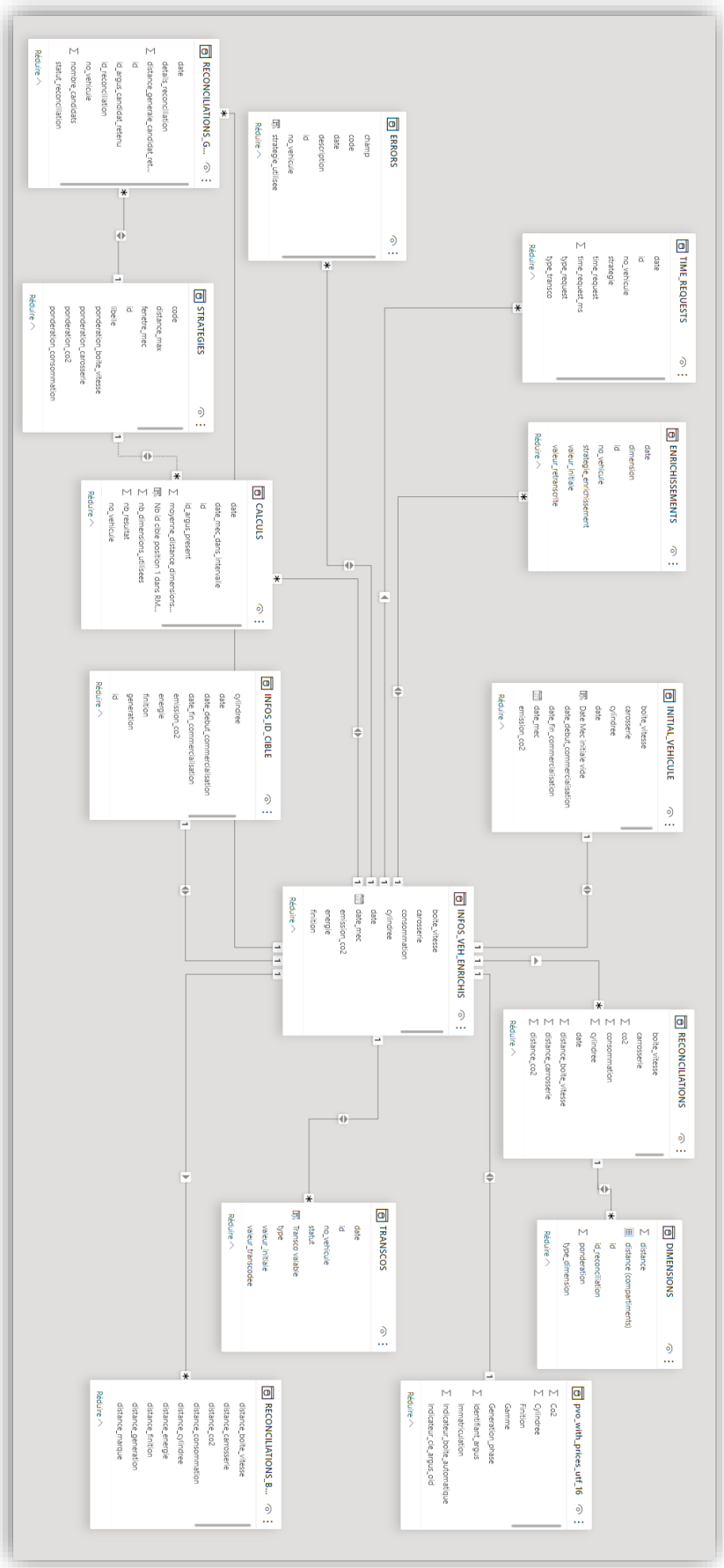
Annexe 7.2.3.a : Interface de la configuration du mapping csv dans l'application lourde



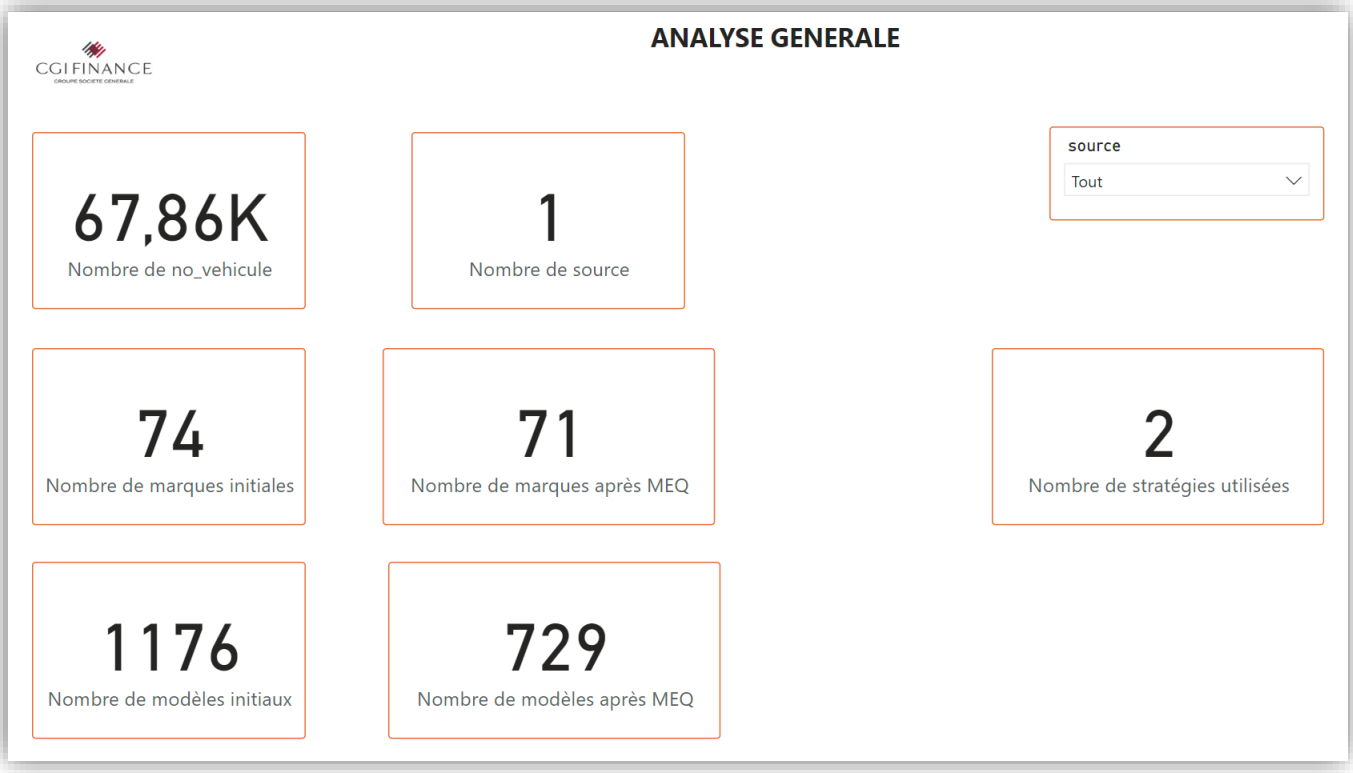
Annexe 7.2.3.b : Interface de l'automatisation des appels RMOD dans l'application lourde



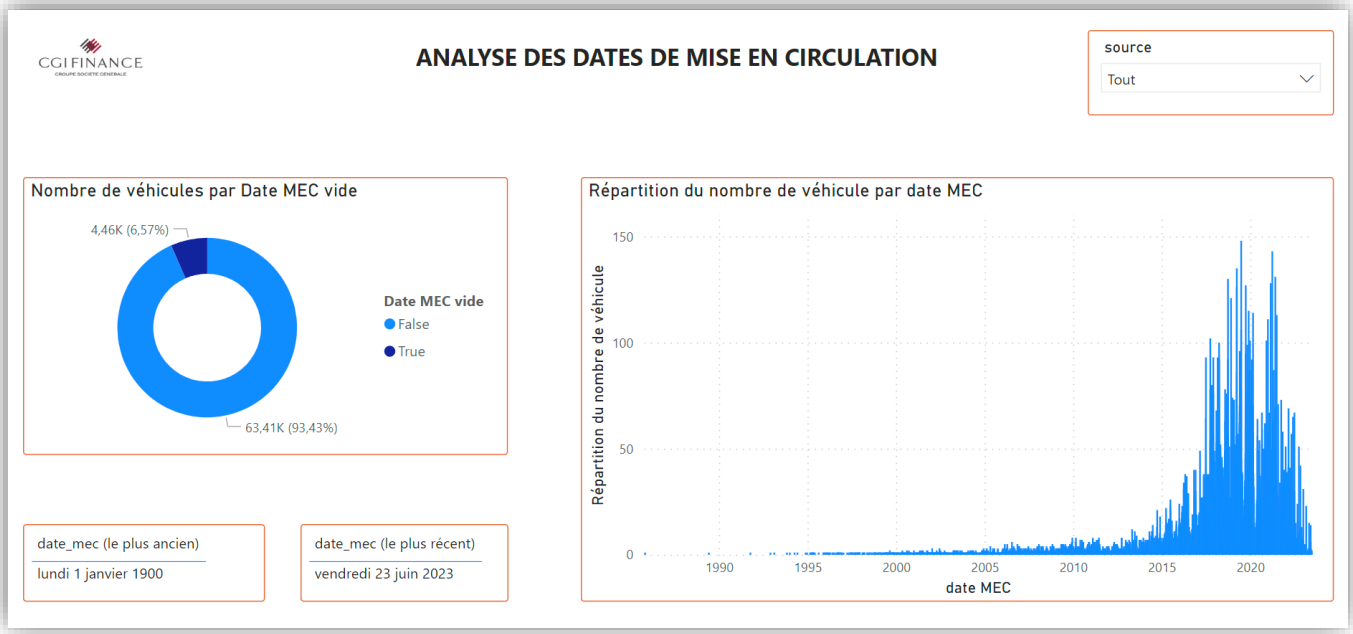
Annexe 7.2.4 : Modèle de dataviz pour l'analyse des résultats de RMOD



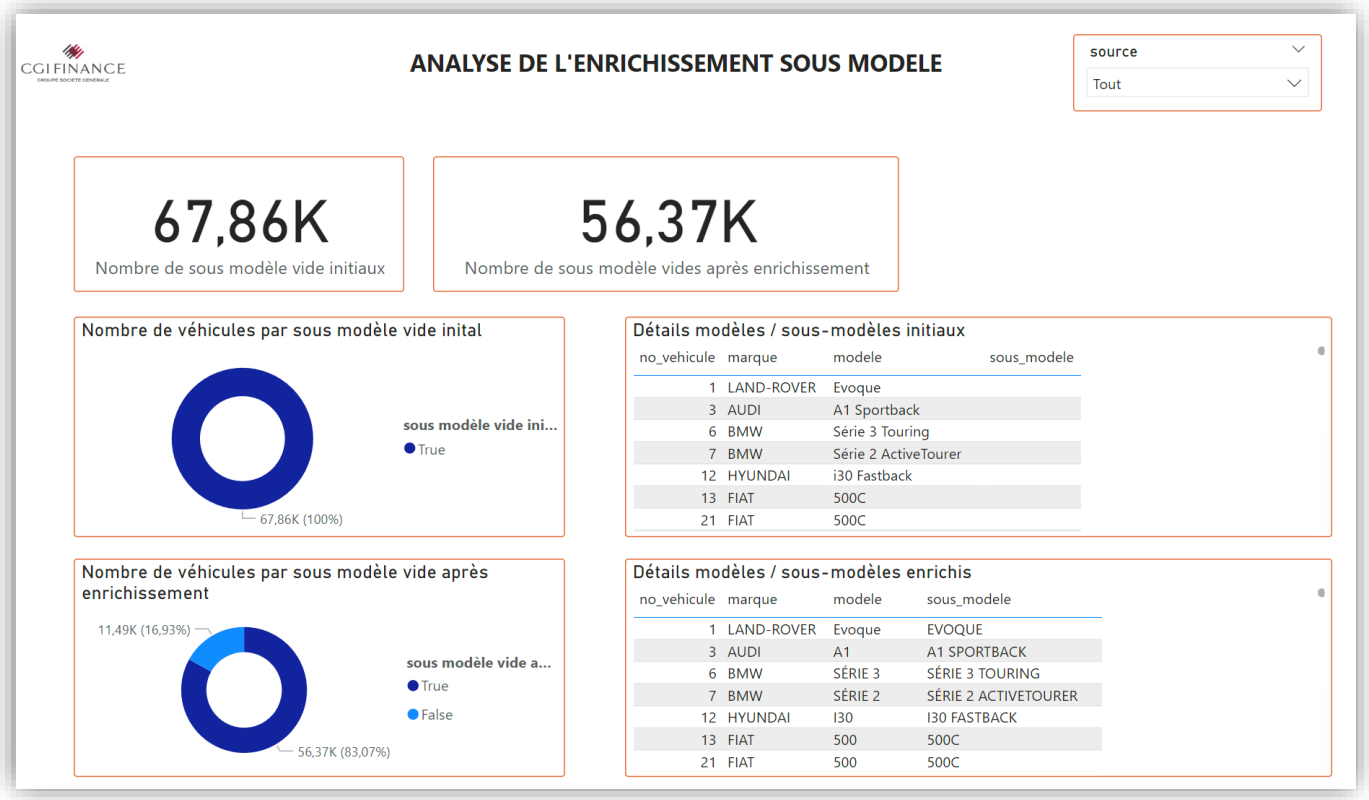
Annexe 7.3.3 : Visuel d'analyse générale



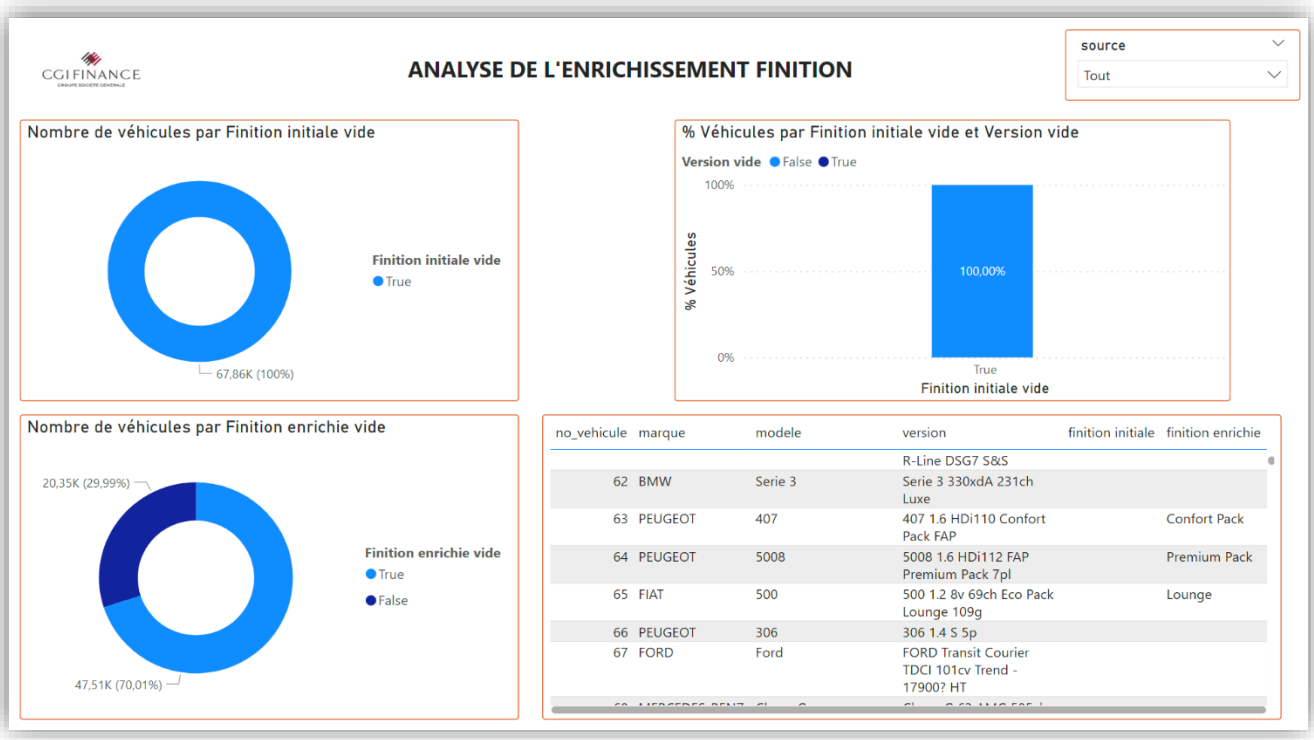
Annexe 7.3.4 : Visuel d'analyse des dates de mise en circulation



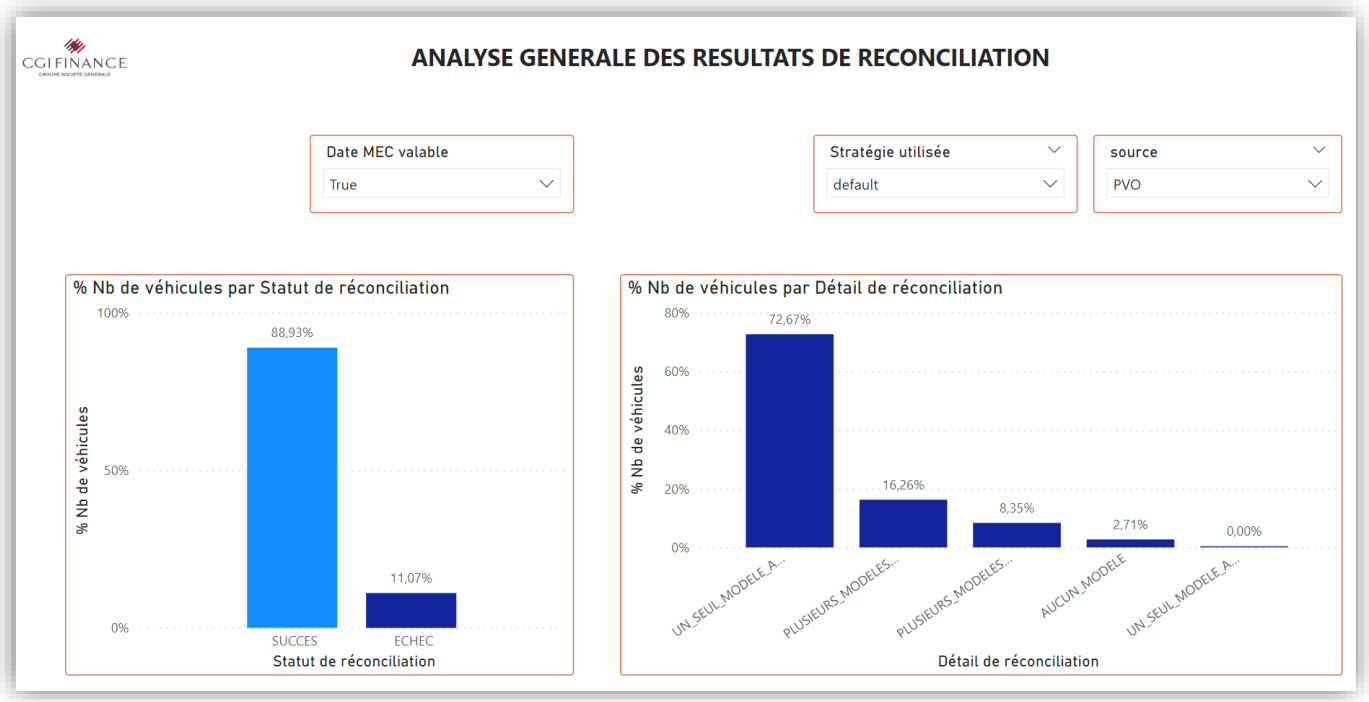
Annexe 7.3.5 : Visuel d'analyse de l'enrichissement sous-modèle



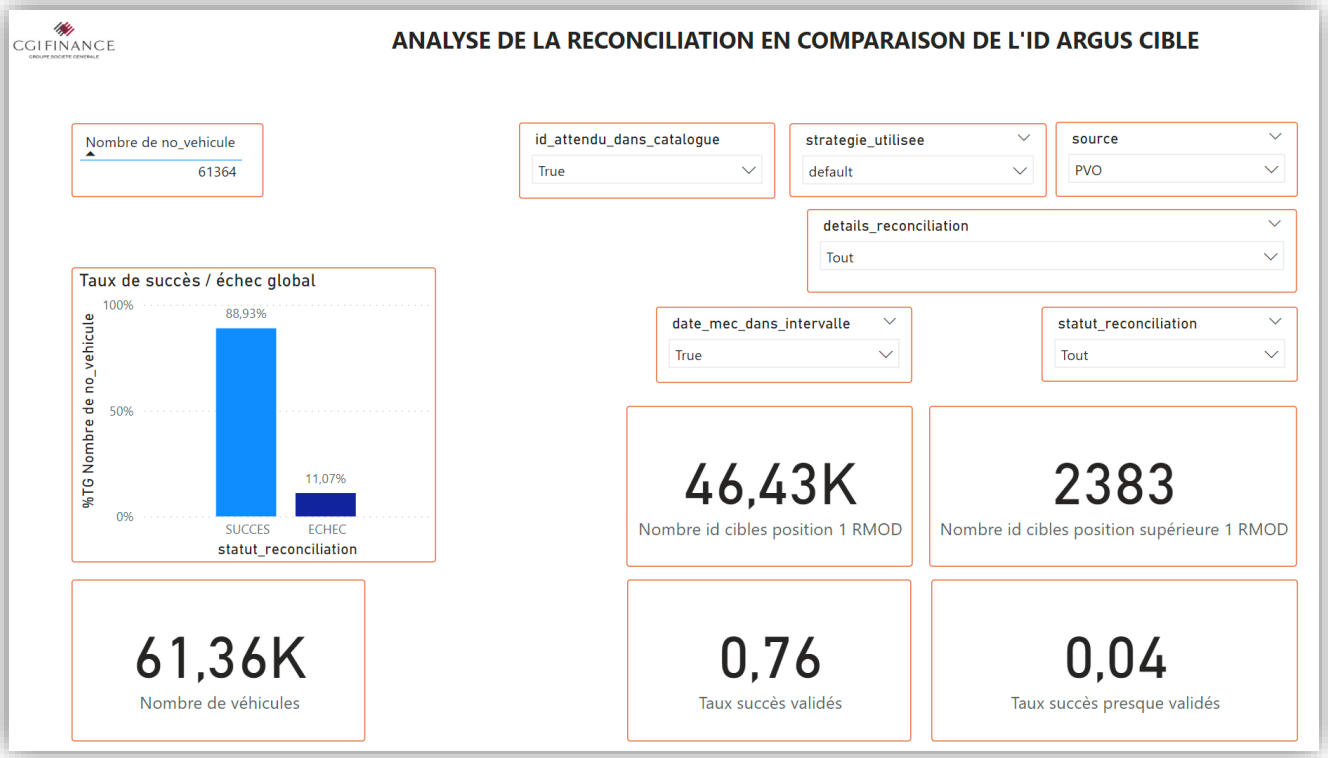
Annexe 7.3.6 : Visuel d'analyse de l'enrichissement de la finition



Annexe 7.3.7 : Visuel d'analyse générale de la réconciliation



Annexe 7.3.8 : Visuel d'analyse de la réconciliation en comparaison avec l'id argus cible



Glossaire / Abréviations

.

.NET 22

Plateforme de développement logiciel créée par Microsoft, utilisée pour construire et exécuter des applications Windows et web.

A

AGILE 63

Méthodologie de gestion de projet qui favorise l'adaptabilité, la collaboration et des cycles de développement courts.

ASP.NET Core 66

Framework open-source pour construire des applications web modernes, développé par Microsoft.

B

Bancassurance 64

Convergence entre les services bancaires et d'assurance, où une banque propose des produits d'assurance.

Batch 34

Processus ou tâche informatique qui s'exécute automatiquement, généralement sans intervention manuelle.

Build 66

Processus de compilation et d'assemblage des composants d'un logiciel pour créer une application exécutable.

C

C.R.U.D. 43

Acronyme pour Create, Read, Update, Delete, désignant les opérations de base sur les données dans une application.

CI/CD 66

Continous Integration/Continuous Deployment, pratiques de développement logiciel pour intégrer et déployer fréquemment des modifications.

CSV 13

Format de fichier **(Comma-Separated Values)** utilisé pour stocker des données tabulaires, où chaque ligne du fichier est un enregistrement de données et chaque enregistrement est séparé par des virgules.

D

Daily Meetings 63

Réunions quotidiennes dans le cadre de la méthodologie AGILE, permettant de coordonner l'équipe sur l'avancement du projet.

DataStage 34

Outil d'intégration de données utilisé pour le traitement et la transformation des données dans un entrepôt de données.

Dataviz 93

Pratique de représentation graphique de données pour faciliter leur compréhension et analyse.

Déploiement 66

Processus de mise en place d'une application ou d'un système informatique dans un environnement de production.

Design Pattern 62

Modèles de conception réutilisables utilisés pour résoudre des problèmes courants en génie logiciel.

DMS 11

(Dealer Management System): Système de gestion utilisé par les concessionnaires automobiles pour gérer leurs opérations commerciales.

E

G

H

J

K

L

M

O

P

R

S

U

V

W

Conception, développement, déploiement et optimisation d'un système pour la réconciliation et la mise en qualité des données de véhicules

Mémoire d'ingénieur C.N.A.M., Valenciennes, 2024

RESUME

Ce mémoire explore le développement de RMOD, un projet ambitieux de CGI Finance, visant à révolutionner la gestion des données de véhicules d'occasion. Au cœur de ce projet se trouve la quête d'une meilleure précision dans l'évaluation des prix des véhicules et une gestion optimisée des informations associées. RMOD se distingue par son approche centrée sur la résolution des problématiques métier et l'efficacité opérationnelle. Il met en lumière la transition de CGI Finance vers des méthodes de travail plus agiles et une utilisation judicieuse des technologies actuelles, tout en générant des économies substantielles. Ce travail détaille non seulement le parcours de conception et de mise en œuvre du projet, mais souligne également l'impact positif de RMOD sur la qualité des données et les processus internes, reflétant une avancée significative dans le secteur de la bancassurance.

Mots-clés : Réconciliation de données, Véhicules d'occasion, Évaluation des prix, CGI Finance, Amélioration des processus, Économie de coûts, Qualité des données, Innovation technologique.

ABSTRACT

This thesis delves into the development of RMOD, an ambitious project by CGI Finance, aimed at revolutionizing the management of used vehicle data. At the heart of this project is the pursuit of improved accuracy in vehicle pricing and enhanced management of related information. RMOD stands out for its focus on resolving business challenges and operational efficiency. It highlights CGI Finance's transition towards more agile working methods and the strategic use of current technologies, while generating substantial savings. This work not only details the journey of designing and implementing the project but also emphasizes the positive impact of RMOD on data quality and internal processes, reflecting a significant advancement in the bancassurance sector.

Keywords : Data Reconciliation, Used Vehicles, Price Evaluation, CGI Finance, Process Improvement, Cost Saving, Data Quality, Technological Innovation.