

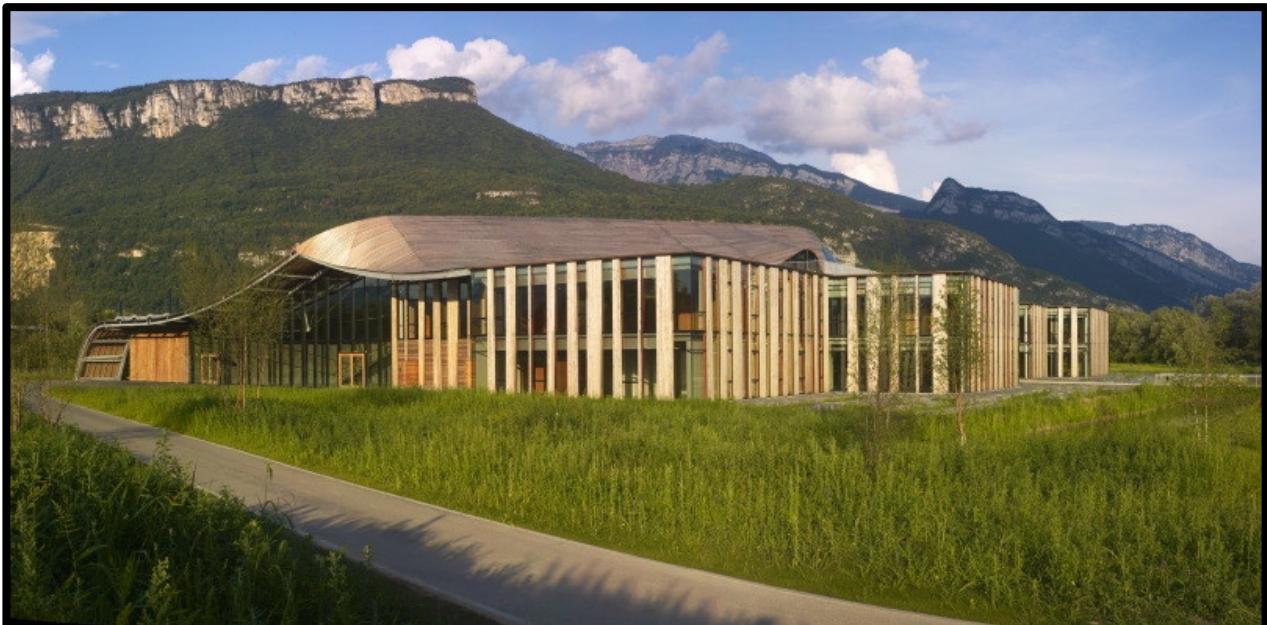


**Rapport de STAGE  
2<sup>e</sup> année BUT SD | Grenoble**

# **Gestionnaire bases de données**

---

**Romain TROILLARD  
SKIS ROSSIGNOL  
8 avril -> 19 juin 2024**



**Rossignol HQ, Moirans**

## Remerciements

Je tiens à remercier toutes les personnes qui ont contribué à la réussite de mon stage chez SKIS ROSSIGNOL. Leur soutien, leurs conseils et leur expertise ont été indispensables et m'ont permis de vivre une expérience extrêmement enrichissante.

Un immense merci à mes tutrices de stage, Mme Stéphanie ROSIN et Mme Gwendoline TONINO, pour leur encadrement, leur disponibilité et leurs précieux conseils tout au long de cette période. Leur soutien constant et leurs orientations m'ont aidé à mieux comprendre les enjeux de mon travail et à progresser rapidement.

Je tiens également à exprimer ma gratitude à Éric CHARRAT pour son aide sur la codification, Quentin PAYEN pour les outils de planification, et Antoine GILLET pour son assistance et ses connaissances à différents niveaux, notamment la navigation dans M3.

Un grand merci à toute l'équipe du service Central Planning pour leur accueil chaleureux et leur aide précieuse. Leur collaboration et leur esprit d'équipe ont rendu mon intégration facile et agréable.

Je suis également reconnaissant envers le département Sciences des Données de l'IUT2 et mes professeurs, en particulier M. Morgan BRASSEL, pour leur soutien académique et leurs enseignements qui m'ont bien préparé pour ce stage.

# Résumé

Stagiaire en tant que **Gestionnaire bases de données** chez Rossignol j'ai été affecté à l'équipe du service **Central Planning**. La plupart des missions de ce service sont effectuées manuellement, elles nécessitent alors beaucoup de temps, d'énergie et peuvent être sujet à des erreurs. Mon objectif était alors d'aider à la réalisation de ces missions et de créer des outils qui permettrait de faciliter le déroulement de ces dernières.

**Mes objectifs de départs portaient sur les sujets suivants :**

- Codification (Gérer/Créer les articles dans l'ERP M3)
- PRICAT (Catalogue des Prix)
- Outils informatiques et automatisation

**Cela m'a amené à réaliser ces travaux :**

- I. Création et modification d'articles dans M3 : Gestion des articles de la ligne produit Footwear, incluant la création de nouveaux articles et la modification des articles existants. [\(2.1.1\)](#)
- II. Développement d'un programme Python pour transformer une Line List modèle en LL SKU : Facilitation de la transformation des modèles en clé unique d'unité (SKU). [\(2.1.2\)](#)
- III. Création d'un programme Python pour identifier les erreurs de correspondance entre les Line Lists et M3 : Vérification et correction des incohérences. [\(2.1.3\)](#)
- IV. Production de PRICAT : Préparation des catalogues de prix pour les revendeurs. [\(2.2.1\)](#)
- V. Outil semi-automatisation PRICAT : Automatisation partielle des catalogues de prix pour améliorer l'efficacité. [\(2.2.2\)](#)
- VI. Développement d'outils pour la planification (avec Quentin PAYEN) : Optimisation des processus de planification pour mieux gérer les dates de planification et les projections des articles. [\(2.3\)](#)

# Summary

My internship at Rossignol as a **database manager** led me to join the **Central Planning** team. The majority of the missions of this service are carried out manually, which requires a lot of time, energy and can lead to errors. My goal was to help carry out these missions and develop tools to facilitate their progress.

## Initially, my goals focused on:

- Coding (Manage/Create items in M3 ERP)
- PRICAT (Price Catalogue)
- Automation and IT tools

## This led me to do this work:

- I. Create and edit articles in M3: Manage articles in the Footwear product line, including creating new articles and editing existing articles. [\(2.1.1\)](#)
- II. Development of a Python program to transform a Model Line List into LL SKU: Facilitating the transformation of models into single key unit (SKU). [\(2.1.2\)](#)
- III. Create a Python program to identify mismatches between Line Lists and M3: Check and correct inconsistencies. [\(2.1.3\)](#)
- IV. Production of PRICAT: Preparation of price catalogues for resellers. [\(2.2.1\)](#)
- V. PRICAT Semi-Automated Tool: Partial automation of price catalogs to improve efficiency. [\(2.2.2\)](#)
- VI. Development of planning tools (with Quentin PAYEN): Optimization of planning processes to better manage planning dates and article projections. [\(2.3\)](#)

# Table des matières

<i>Remerciements</i>	<b>2</b>
<i>Résumé</i>	<b>3</b>
<i>Summary</i>	<b>4</b>
<i>Table des matières</i>	<b>5</b>
<i>Table des références</i>	<b>6</b>
<i>Glossaire</i>	<b>7</b>
<b>1. Introduction</b>	<b>8</b>
1.1. Présentation de l'entreprise	8
1.2. Objectifs et problématique	8
1.3. Existant	9
<b>2. Missions</b>	<b>10</b>
<b>2.1. Codification</b>	<b>10</b>
2.1.1. Création/Modification d'articles	10
2.1.2. Transformation d'une Line List modèle en Line List SKU	11
2.1.3. Vérification correspondance entre Line List (PLM) et M3 (ERP)	14
<b>2.2. PRICAT</b>	<b>18</b>
2.2.1. Construction manuelle d'un PRICAT	19
2.2.2. Outil semi-automatisation en VBA	20
<b>2.3. Outils Planification</b>	<b>22</b>
2.3.1. ITP004	22
2.3.2. ITP013	23
2.3.3. ITP015	24
2.3.4. ITP014 & ITP016	24
2.3.5. ITP017	25
2.3.6. Bilan ITP	25
<b>3. Conclusion</b>	<b>26</b>
3.1. Intérêt méthodologique et pratique	26
3.2. Bilan personnel	26

# Table des références

## Table des schémas

Schéma 1 : Fonctionnement global du programme "SKU_FROM_LL.py"	11
Schéma 2 : Fonctionnement global du programme "CHECK_M3_LL"	14
Schéma 3 : Plage créée à partir du choix de saison de l'utilisateur	15
Schéma 4 : Fonctionnement global PRICAT	18
Schéma 5 : Échange EDI	19
Schéma 6: Principales entrées et sorties de la plateforme centrale de Rossignol	23

## Table des codes

Code 1 : Création du Dataframe des SKU à partir de la liste des tailles des modèles	12
Code 2 : Manipulation des tailles afin d'obtenir le code sous le bon format (si longueur inférieure à 3 -> des 0 devant)	12
Code 3 : Création du Dataframe des échantillons (Samples)	13
Code 4 : Export des Dataframes dans 2 feuilles d'un même fichier Excel	13
Code 5 : Récupération des PATH des fichiers sélectionnés dans un explorateur	15
Code 6 : Création du Dataframe "Base Articles" à partir d'une requête SQL	15
Code 7 : Regroupement des SKU des mêmes modèles avec les mêmes erreurs	16
Code 8 : Utilisation de la fonction "create_mdi_mms001"	16
Code 9 : Utilisation de "COALESCE" dans une requête SQL	25
Code 10 : fonction "create_model_color"	34
Code 11 : fonction "chaine_en_liste"	34
Code 12 : fonction "create_seasons_range"	35
Code 13 : fonction "extract_brand_code"	35
Code 14 : Fonction "create_mdi_mms001"	36

## Table des images

Image 1 : Feuille "USER"	20
--------------------------	----

## Table des documents (en annexe)

Document 1 : Guide Utilisateur "SKU_FROM_LL.py"	27
Document 2 : Guide Développeur "SKU_FROM_LL.py"	29
Document 3 : Guide Utilisateur "check_M3_LL.py"	30
Document 4 : Guide Développeur " check_M3_LL.py"	32

# Glossaire

**SKU (Single Key Unit)** : Identifiant de produit unique codé de la façon suivante :

- Code modèle (7 caractères) + 2 espaces + code couleur (3 caractères) + taille (3 caractères)

Exemple : DANWE01 000070

**PRICAT (Prix Catalogue)** : Fichier dans lequel on répertorie la liste des articles et toutes ses informations (prix de gros, prix de vente, taille, genre ...) pour un revendeur comme INTERSPORT.

**Line List (LL)** : Fichier Excel issu du PLM/KUBIX répertoriant les modèles avec leur couleur et taille à inclure dans l'ERP M3.

**Forecast** : Prévisions commerciales.

**ERP M3** : l'ERP (Entreprise Resource Planning) est un système intégré de gestion des ressources, il permet de centraliser différentes fonctions commerciales. M3 fait référence à un système ERP spécifique, ce système est principalement utilisé dans les secteurs de la fabrication, de la distribution, et de la gestion de chaîne d'approvisionnement. Il permet la gestion des achats, des ventes, de la production, de la gestion des stocks et des capacités. L'ERP M3 est donc au cœur de l'entreprise.

**MDI (M3 Data Import)** : Interface permettant la création et la modification en masse dans M3 à partir d'un fichier CSV et d'un "Mapping MDI" (On paramètre les champs à modifier).

**BO** : Suite logicielle BI permettant d'analyser, de visualiser, et de partager différentes informations notamment issues de M3 (dans le cas de Rossignol).

**ITP** : Projet visant à améliorer certains aspects de la planification pour livrer dans les temps, ne pas faire face à un mur d'expéditions à la plateforme centrale de Saint-Étienne-de-Saint-Geoire.

**OD** : L'Ordre de Distribution fait référence à la livraison d'une commande entre les dépôts / usines ferme

**POD** : La Proposition d'Ordre de Distribution correspond à une livraison d'une commande entre dépôts / usines non ferme

**OA** : L'Ordre d'Achat est une commande vers un fournisseur ferme

**POA** : La Proposition d'Ordre d'Achat est une commande vers un fournisseur qui n'est pas encore ferme

# 1. Introduction

## 1.1. Présentation de l'entreprise

SKIS ROSSIGNOL est une entreprise française créée en 1907 spécialisée dans la fabrication de skis, de chaussures de ski, de vêtements et d'accessoires pour les sports d'hiver. Elle compte 1230 employés dans le monde dont 680 en France. Elle distribue ses produits à l'international dans 52 pays. Rossignol Group possède aussi d'autres marques importantes et essentielles à son développement :

- Dynastar (skis, équipements haut de gamme pour tous les niveaux)
- Lange (chaussures de ski)
- Look (fixation)

En plus de son expertise dans les sports d'hiver, SKIS ROSSIGNOL explore également la création d'autres types d'articles comme des chaussures, des équipements, élargissant ainsi sa gamme de produits pour répondre aux besoins variés des amateurs de sports et d'activités en plein air tout en conservant son ADN : la montagne.

Skis Rossignol compte 4 usines en Europe dont 2 en France :

- Sallanches : production, au pied du Mont-Blanc, de skis alpins haut de gamme en petite et moyenne séries pour Rossignol et Dynastar et de skis pour les gammes Junior.
- Nevers : fixations sous la marque Look, N°2 mondial.
- Artés (Espagne) : site dédié à la production des skis alpins en noyau bois, de technologie traditionnelle pour les marques Rossignol et Dynastar et les skis de fond haut de gamme.
- Montebelluna (Italie) : chaussures alpines haut de gamme sous les marques Rossignol et Lange, patins à glace haut de gamme Risport. Centre d'expertise en conception/design pour les chaussures (incluant le Footwear Rossignol).

De plus elle a aussi :

- Un entrepôt logistique central basé à Saint-Etienne-de-Saint-Geoire en Isère.
- Un hub global Apparel & D2C en Italie à Milan.

## 1.2. Objectifs et problématique

Mes objectifs en tant que **gestionnaire de bases de données** étaient d'aider à la réalisation des tâches courantes comme créer et gérer les articles dans l'ERP M3 (codification), de préparer des catalogues de prix (PRICAT) pour les magasins souhaitant vendre des produits provenant de chez Rossignol. Ces missions étant totalement manuelles, je devais réfléchir à comment automatiser ou faciliter le déroulement de ces dernières. Enfin j'ai été sollicité pour créer des outils en lien avec les deux sujets précédents mais aussi pour l'équipe des planificateurs.

### **1.3. Existant**

A mon arrivée j'ai découvert quelles étaient les données avec lesquelles travaille Rossignol, ces données sont stockées dans l'ERP M3 qui est au cœur de l'entreprise. Les données de l'entreprise sont alors de tout type (du code article à des dates en passant par des informations textes et du stock chiffré). De plus pour bon nombre de sujets il existait une procédure à suivre (certaine avec des approximations). Enfin certains outils Excel pour la Supply Chain étaient mis en place, ce ne sont pas des outils que j'utilise en tant qu'utilisateur mais j'ai été amené en tant que développeur à travailler sur certains.

## 2. Missions

### 2.1. Codification

L'une de mes principales missions a été la codification, cela consiste à gérer les articles dans l'ERP M3. Tout d'abord comment sont codés les saisons chez Rossignol ?

1. Fall-Winter (FW) qui commence le 1<sup>er</sup> Septembre et finit le 31 Mars (Le nombre avant le code 'FW' est celui de l'année de fin (01/09/2024 -> 31/03/2025 = FW25)
2. Spring-Summer (SS) qui commence 1<sup>er</sup> Avril et finit le 31 Aout

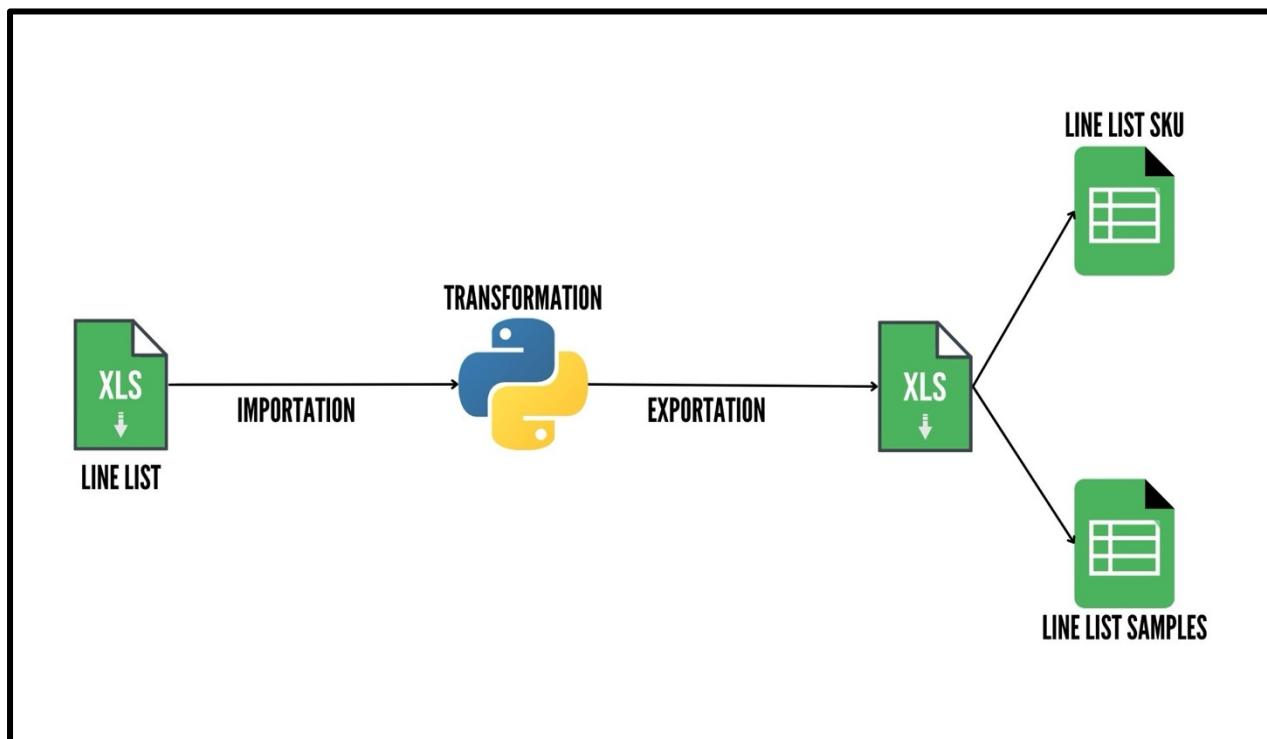
#### 2.1.1. Création/Modification d'articles

Mon domaine concernait les articles de la ligne produit Footwear. Ainsi je devais créer les articles qui étaient nouveaux (valeur « N New » dans la colonne Lyfecycle), et effectuer des potentielles modifications sur les modèles reconduits (valeur « C Carryover) comme la dernière année de commercialisation qui passait de 25SS à 26FW ou encore un changement de genre sur le produit (Unisex au lieu de Men) et d'autres. Pour la création des nouveaux articles je devais d'abord créer le modèle de l'article à partir d'un autre article existant en copiant ce dernier et en effectuant les quelques modifications nécessaires comme le nom et le code du produit. Ensuite je devais affecter une grille de taille et une grille de couleurs au modèle afin de créer les [SKU](#) depuis un menu qui fait une matrice tailles/couleurs. Par exemple en mettant un « 1 » dans le couple F11 (couleur) et 075 (taille) je créais à partir du modèle RNNMA50 le SKU : RNNMA50 F11075. Enfin je devais créer une entité pour chaque modèle couleur (ENTRNNMA50 F11 par exemple), ces entités sont utilisées par la planif pour les [forecasts](#).

Au final j'ai pu créer plusieurs SKU que ce soit pour le footwear ou encore pour le club Med (kits de skis d'occasion). Pour ce qui s'agit des difficultés rencontrées, du point de vue technique rien à signaler mais au niveau de l'intérêt c'était plus compliqué car c'est un travail redondant, on ne sait pas vraiment quel est l'article qu'on créer (pas de photo mais cela est normal car l'article peut être annulé avant le lancement de la saison et la codification des SKU est fait environ 15 mois avant le début de la saison). De plus la partie de création des entités renforce cette impression de redondance.

## 2.1.2. Transformation d'une Line List modèle en Line List SKU

Schéma 1 : Fonctionnement global du programme "SKU\_FROM\_LL.py"



Il faut savoir que les Line List sont sous la forme modèle, cela veut dire que nous avons pour chaque ligne un modèle avec sa couleur et toutes ses tailles (ainsi que de nombreuses autres informations non essentielles pour le raisonnement) :

Tableau 1 : Extrait d'une ligne de "Line List"

Product Code	Product Name	Color	...	Enable Size
RLNMIJ78	SIDELHORN PKB JKT	41P GOLDEN GATE		0XS, 00S, 00M, 00L, 0XL, 2XL, 3XL

Je devais alors créer un programme permettant d'obtenir une LL sous la forme SKU :

Tableau 2 : Extrait du résultat attendu (LL SKU)

SKU	Product Code	Product Name	Color Code	Color name	...
RLNMIJ78 41P0XS	RLNMIJ78	SIDELHORN PKB JKT	41P	GOLDEN GATE	
RLNMIJ78 41P00S	RLNMIJ78	SIDELHORN PKB JKT	41P	GOLDEN GATE	
...	...	...	...	...	
RLNMIJ78 41P3XL	RLNMIJ78	SIDELHORN PKB JKT	41P	GOLDEN GATE	

J'ai eu la chance de pouvoir réaliser cette mission comme je l'entendais, avec pour seule contrainte que le résultat soit conforme. Naturellement, je me suis tourné vers Python, car c'est le langage que je maîtrise le mieux et il est relativement puissant pour le traitement de données. J'ai alors utilisé le package pandas, qui me permet de manipuler et nettoyer les données issues de fichiers Excel sous forme de DataFrame.

Pour commencer, après import du fichier Excel, j'ai créé les colonnes "Color name" et "Color code" à partir de la colonne "Color". Puis création du modèle couleur grâce à la fonction "create\_model\_color" qui crée une chaîne de 12 caractères à partir du modèle et du code couleur, de plus transformation des valeurs dans la colonne "Enable Size" en liste grâce à la fonction "chaine\_en\_liste" qui utilise la méthode `.split()`. J'ai découvert une méthode que je ne connaissais pas auparavant : `.explode()`. Cette méthode permet d'éclater une colonne (ici "Enable Size") où les valeurs sont une liste ou un tuple, et de créer une ligne pour chaque élément de la liste tout en conservant les informations des autres colonnes.

#### *Code 1 : Création du Dataframe des SKU à partir de la liste des tailles des modèles*



```
df_LL_SKU = df_LL.explode('enable_size').reset_index(drop=True)
```

J'ai ensuite manipulé cette nouvelle colonne "Enable Size", qui contient désormais un seul élément par ligne, pour que chaque valeur ait une longueur de 3 caractères (par exemple, convertir "S" en "00S"). Pour cela, j'ai utilisé la méthode `.zfill(3)` qui complète les chaînes de caractères avec des zéros au début jusqu'à ce qu'elles atteignent une longueur de 3 caractères.

#### *Code 2 : Manipulation des tailles afin d'obtenir le code sous le bon format (si longueur inférieure à 3 -> des 0 devant)*



```
df_LL_SKU['enable_size'] = df_LL_SKU['enable_size'].str.zfill(3)
```

Pour terminer, j'ai créé la colonne "SKU" à partir des colonnes "Model color" et "Enable Size". Enfin j'ai sélectionné et réorganisé les colonnes utiles et exporter le DataFrame au format ".xlsx".

Les Line List sous forme SKU permettront de générer une base de données article détaillée afin de pouvoir travailler les données inter-systèmes.

Après cela on m'a demandé de créer une feuille pour visualiser les échantillons (samples), pour les nouveaux modèles couleur il est nécessaire de créer un échantillon (une taille) qui est codé selon la norme suivante : Modèle + E + espace + couleur + taille, on remarque que c'est la même codification que pour les SKU mais au lieu du premier espace on met un 'E'. J'ai alors sélectionné les nouveaux modèles couleur et créer un DataFrame des échantillons.

### Code 3 : Création du Dataframe des échantillons (Samples)

```
check_master = (df_LL_SKU['lifecycle'] == 'N New') & (df_LL_SKU['sample size'] == df_LL_SKU['enable size'])
df_LL_SKU['master'] = np.where(check_master, 'X', '')

df_LL_sample = df_LL_SKU[df_LL_SKU['master'] == 'X']
df_LL_sample.loc[:, 'product code'] = df_LL_sample['product code'] + 'E'
df_LL_sample.loc[:, 'model col'] = df_LL_sample.apply(create_model_color, axis=1)
df_LL_sample.loc[:, 'sku'] = df_LL_sample['model col'] +
df_LL_sample['enable size']
```

Enfin j'exporte les deux DataFrames dans UN fichier Excel (une feuille pour chaque DataFrame) qui sera dans le dossier 'Téléchargement' grâce à la méthode `.expanduser()` du module `os.path`.

### Code 4 : Export des Dataframes dans 2 feuilles d'un même fichier Excel

```
file_name = input("Choisir un nom de fichier : ")
DOWNLOADS_PATH = os.path.join(expanduser('~'), 'Downloads')
with pd.ExcelWriter(os.path.join(DOWNLOADS_PATH, f"{file_name}.xlsx")) as writer:
    df_LL_SKU.to_excel(writer, sheet_name='SKU COLLECTION', index=False)
    df_LL_sample.to_excel(writer, sheet_name='SKU SMS', index=False)
```

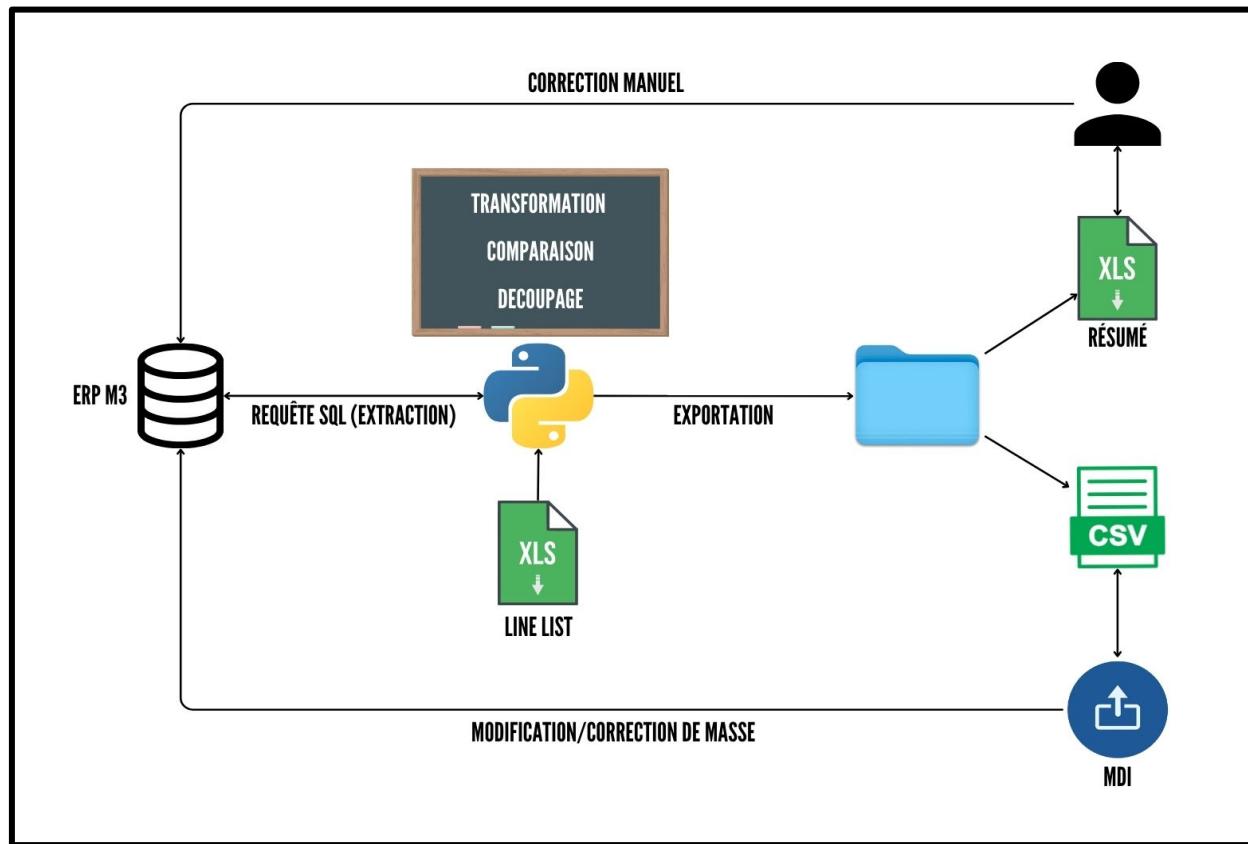
Je n'ai fait face à aucune difficulté à tous les niveaux lors de la réalisation de ce programme.

Un guide utilisateur et développeur sont disponibles en annexe :

- DOCUMENT 1 : GUIDE UTILISATEUR "SKU\_FROM\_LL.PY"
- DOCUMENT 2 : GUIDE DEVELOPPEUR "SKU\_FROM\_LL.PY"

### 2.1.3. Vérification correspondance entre Line List (PLM) et M3 (ERP)

Schéma 2 : Fonctionnement global du programme "CHECK\_M3\_LL"



Lors de la codification, il arrive que des erreurs soient commises surtout sur des articles où une information a changé par rapport à la saison précédente mais ces changements ne sont pas explicitement notifiés. Auparavant après avoir codifié l'ensemble des Line List (LL) d'une saison, la procédure était d'imprimer ces dernières et de comparer ligne par ligne avec ce qui était dans l'ERP M3, cette étape prenait énormément de temps et on n'était toujours pas à l'abri qu'il y ait des incohérences. Alors ma mission était de développer un outil pouvant comparer une LL avec M3. Comme pour le programme précédent j'avais le choix de comment j'allais réaliser la mission donc encore une fois je me suis orienté vers Python. J'ai commencé par établir quels étaient les problèmes de correspondance les plus fréquents et j'ai répertorié 8 champs :

- Nom
- Marque (Rossignol, Dynastar, Lange etc)
- Ligne Produit (exemple S1 pour Footwear)
- Origine (d'où vient le produit)
- SMU (Savoir si c'est pour des clients particuliers ou non)
- Genre
- Saison (La dernière saison sur lequel l'article doit être actif 26FW par exemple)
- Autre (Article pas créé, absent de la sélection SQL car saison trop éloignée ...)

Mon programme propose alors à l'utilisateur de sélectionner une ou plusieurs LL de la même saison grâce au module `tkinter` qui permet d'ouvrir un explorateur de fichier, et après sélection des fichiers, créer une liste des chemins. ([CODE 5](#))

*Code 5 : Récupération des PATH des fichiers sélectionnés dans un explorateur*

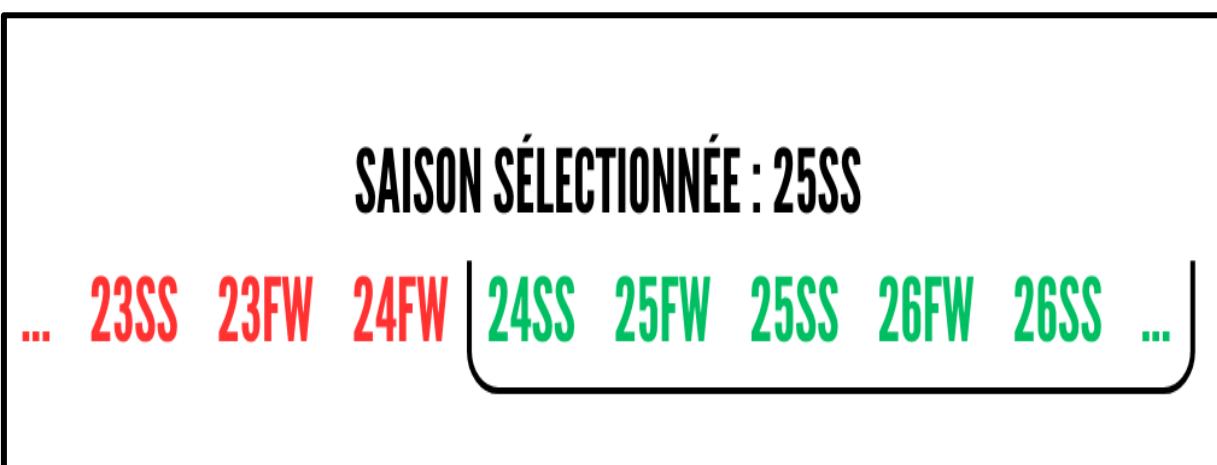


```
LL_files = filedialog.askopenfilenames(title='Sélectionner les LINE LIST',  
multiple=True)
```

Chaque LL est parcouru en vérifiant si tous les champs essentiels sont présents. Si c'est le cas on ajoute la LL à une liste où il y aura tous les Dataframe (1 par LL), puis concaténation des LL afin d'avoir un seul objet sur lequel itéré.

L'utilisateur est alors amené à rentrer la saison des LL, afin que la requête SQL aille chercher dans une plage de saisons allant de : saison sélectionnée – 2 saisons. ([CODE 12](#))

*Schéma 3 : Plage créée à partir du choix de saison de l'utilisateur*



Pour comparer avec les articles présents dans M3 on créer un DataFrame à partir d'une requête SQL grâce au lien ODBC qui permet d'accéder à la base de l'ERP qui fera office de 'Base Articles'. ([CODE 6](#))

*Code 6 : Création du DataFrame "Base Articles" à partir d'une requête SQL*



```
M3Query = """SELECT ... FROM ... WHERE ..."""
df_M3_BA = pd.read_sql(M3Query, engine)
# engine est le lien ODBC avec M3
```

Ensuite on filtre le DataFrame des LL afin de ne garder que les articles actifs et enlever ceux qui ont été annulé ('X' dans le champ 'Drop'). Le début du programme est très

similaire à celui de création des SKU, c'est-à-dire nettoyage des données, création de colonnes, LL sous la forme de SKU ...

Après cela on crée un DataFrame vide pour cibler les problèmes (une colonne pour chaque problème "BRAND PROBLEM", "GENDER PROBLEM" etc...) puis on parcourt tous les SKU des LL et on cherche leur correspondance dans la base articles et on compare les 8 champs principaux sachant que la valeur présente dans la LL est là référence, dès lors pour chaque champ si dans M3 ce n'est pas égal au même champ que dans la LL, on ajoute dans la colonne référençant le problème du dictionnaire temporaire la valeur suivante :

**LL : \*valeur dans la LL ||| M3: \*valeur dans M3**

Si une des colonnes "PROBLEM ..." n'est pas vide on ajoute le dictionnaire temporaire au DataFrame des problèmes. On obtient alors un DataFrame complet avec les erreurs sur les SKU, mais les erreurs sont souvent sur le modèle directement donc je découpe le DataFrame en deux autres :

- Un pour les modèles (quand 2 SKU ou + du même modèle ont le/les mêmes problèmes)
- Un pour les SKU

*Code 7 : Regroupement des SKU des mêmes modèles avec les mêmes erreurs*



```
df_model_problem = df_SKU_problem.groupby(['MODEL', 'M3 NAME', 'NAME PROBLEM', 'BRAND PROBLEM', 'GROUP PROBLEM', 'SUPPLIER PROBLEM', 'ORIGIN PROBLEM', 'SMU PROBLEM', 'GENDER PROBLEM', 'SEASON PROBLEM', 'OTHER PROBLEM']).agg({
    'SKU': list,
    'LL FILE': 'last'
}).reset_index()
```

Enfin je créer des fichiers CSV compatible avec les mappings MDI sur les problèmes comme : Genre, Origin, SMU, Ligne Produit, Saison. Cela permettra de corriger ces erreurs plus rapidement. [\(CODE 14, CODE 8\)](#)

*Code 8 : Utilisation de la fonction "create\_mdi\_mms001"*



```
mdi_maj_gender = create_mdi_mms001('GENDER PROBLEM', 'CFI5',
dico_correspondance_gender)
mdi_maj_smu = create_mdi_mms001('SMU PROBLEM', 'CFI3',
dico_correspondance_smu)
mdi_maj_group = create_mdi_mms001('GROUP PROBLEM', 'ITCL')
mdi_maj_season = create_mdi_mms001('SEASON PROBLEM', 'CFI4')
```

Le programme finit son exécution en enregistrant dans un dossier placé dans 'Téléchargement' un fichier Excel avec une feuille pour les erreurs sur les modèles et une autre feuille pour les SKU et bien sur l'ensemble des fichiers CSV, si aucune erreur est détectée rien n'est enregistré et un message indique à l'utilisateur que tout est OK.

Voici un aperçu de ce que retrouve l'utilisateur dans le fichier Excel :

Tableau 3 : Extrait de la feuille "MODEL" qui répertorie les erreurs qui sont au niveau du modèle

MODE L	M3 NAM E	NAME PROBL EM	BRAND PROBL EM	GROUP PROBL EM	ORIGIN PROBL EM	SMU PROBL EM	GENDER PROBL EM	OTHER PROBL EM
RENN7 01	EXP JUNI OR				LL : HK M3 : CN			
RENT8 01	CRU MB			LL : A06 M3 : A03			LL : Men M3 : Unisex	

On voit ici qu'il faudra modifier pour 'RENT801' le groupe en mettant 'A06' et le genre en mettant 'Men' au lieu de 'Unisex'

Tableau 4 : Extrait de la feuille "SKU" qui répertorie les erreurs au niveau du SKU

SKU	M3 NAME	NAME PROBL EM	BRAND PROBL EM	GROUP PROBLE M	ORIGIN PROBLE M	SMU PROBLE M	GENDER PROBLE M	OTHER PROBLE M
RVEW4 15 0000TU	BATTL E			LL : AO3 M3 : ZZZ				

La colonne "OTHER PROBLEM" indique que le(s) SKU n'a pas été trouvé dans l'extraction SQL M3, cela implique que soit il n'existe pas et il faut donc le créer, soit sa saison commerciale n'était déjà pas conforme sur la LL de l'année d'avant (et donc elle n'est pas dans la plage des saisons utilisé lors de la requête SQL).

En conclusion ce programme est la bienvenue du fait qu'il identifie clairement les erreurs (car vérifier tous les articles à la main implique aussi de vérifier les articles sans erreur (la majorité)), de plus les fichiers CSV combiné à MDI permettent une modification de masse plus rapide. Par conséquent cet outil permet un gain de temps considérable, et permet de ne pas passer à côté des erreurs commises. Tout comme le programme de transformation de LL modèle en SKU, aucune difficulté particulière à remonter.

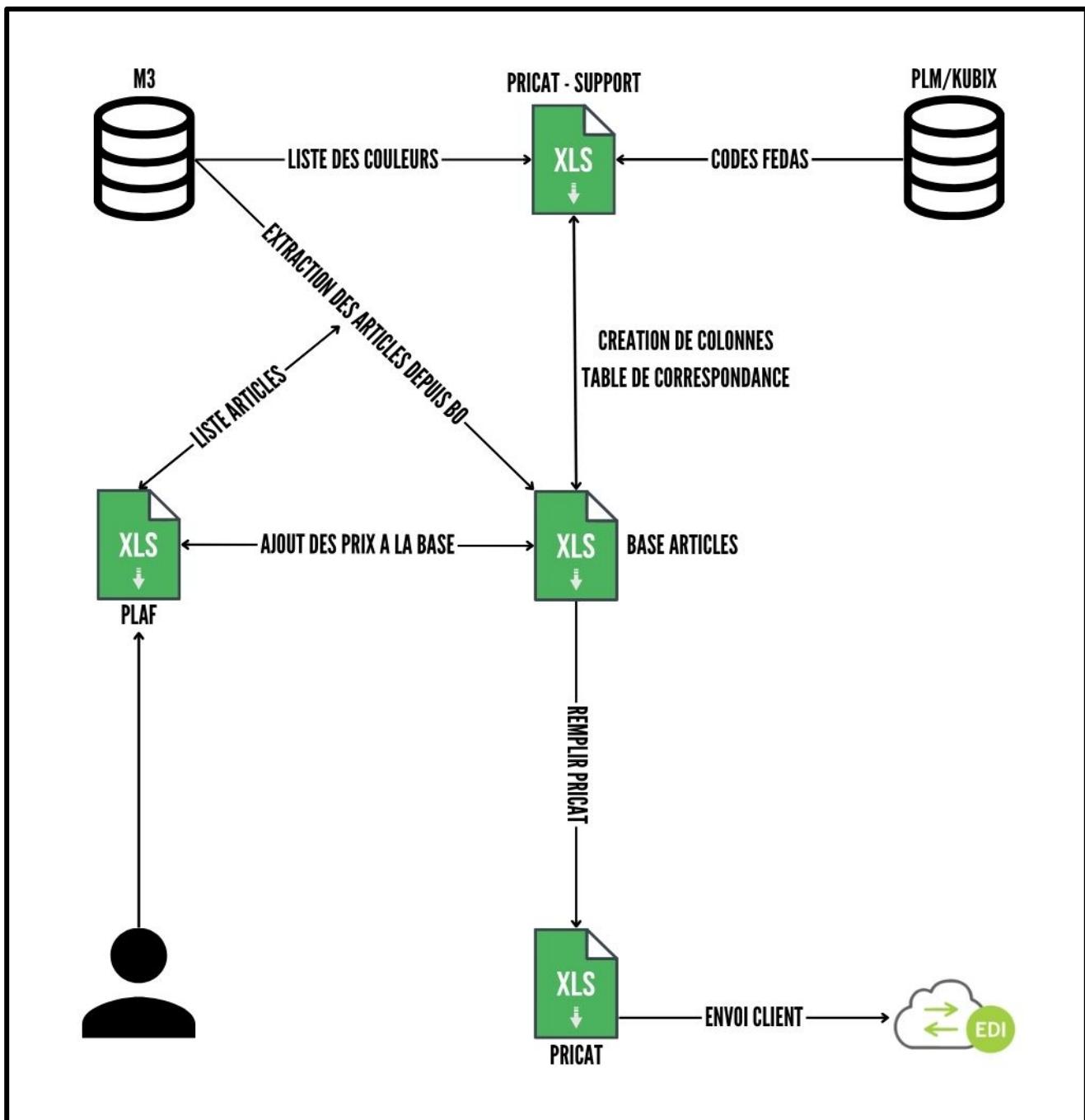
Un guide utilisateur et développeur sont disponibles en annexe :

- DOCUMENT 3 : GUIDE UTILISATEUR "CHECK\_M3\_LL.PY"
- DOCUMENT 4 : GUIDE DEVELOPPEUR " CHECK\_M3\_LL.PY"

## 2.2. PRICAT

Le sujet PRICAT a été une mission plus compliquée à gérer pour moi, dû au fait que les données proviennent de différentes sources et qu'elles ne soient pas toujours disponible.

Schéma 4 : Fonctionnement global PRICAT



### 2.2.1. Construction manuelle d'un PRICAT

Tout d'abord comment est construit un PRICAT ? Les magasins type SPORT2000 ou encore INTERSPORT font une demande de catalogue avec les prix pour qu'ils puissent ensuite acheter et revendre dans leurs magasins les articles de ROSSIGNOL. On récupère alors les informations essentielles comme la saison, le pays du magasin etc pour ensuite demander au "Country Manager" du pays la liste des produits avec les prix de gros et de reventes à renseigner dans le PRICAT. Il est aussi important d'identifier grâce à la base donnée client comment va être envoyé le PRICAT, il y a 2 méthodes, soit un simple fichier Excel ou alors par EDI pour les magasins partenaires.

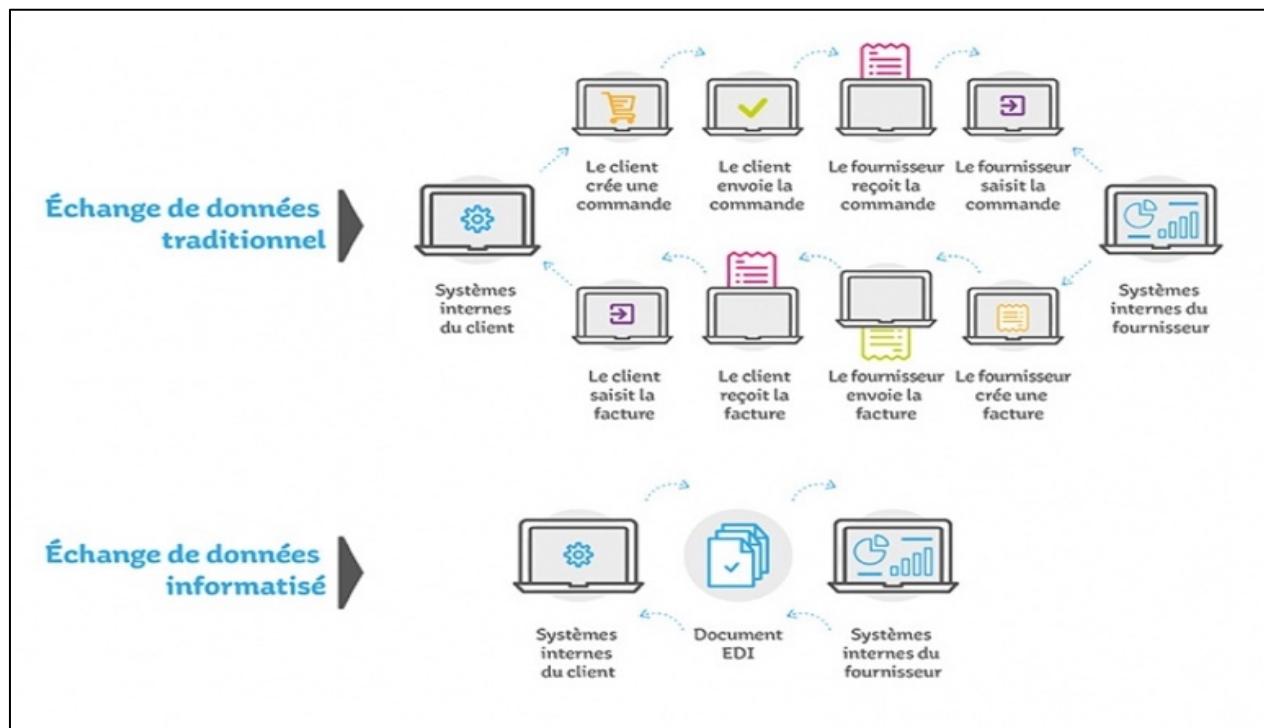


Schéma 5 : Échange EDI

EDI permet de remplacer les échanges physiques de document entre les entreprises, les données sont envoyées sous un format standardisé.

Tous les documents commerciaux sont donc dématérialisés que ce soit les bons de commandes, les factures, les paiements, les PRICAT etc ...

Ensuite à partir de la fiche des articles avec leur prix, on va récolter un bon nombre d'informations concernant les articles grâce à un export BO. Le reste des informations sera obtenue en effectuant des RECHERCHEX entre les informations déjà présente grâce à la requête BO et un fichier Excel "PRICAT – SUPPORT". Après avoir obtenue l'ensemble des informations, on construit le PRICAT et on peut envoyer au client le catalogue.

La fabrication d'un PRICAT n'est pas quelque chose de très compliqué du point de vue technique mais c'est au niveau du cheminement que les choses deviennent plus durs, il est fréquent qu'entre la demande de PRICAT et la réception de la fiche des articles avec les prix se passent des jours voire des semaines, de plus lors de la collecte des données il n'est pas impossible d'avoir des données manquantes et l'endroit où chercher l'information n'est pas vraiment très clair.

## 2.2.2. Outil semi-automatisation en VBA

Malgré tout j'ai développé un outil semi-automatiser afin d'accélérer la production d'un PRICAT. Pour ce faire je n'ai eu d'autre choix que de travailler sur Excel et donc en VBA. Le programme ne fonctionne que dans le cas d'un échange par EDI (ce qui représente 90% des demandes), il est composé de 4 feuilles :

- Template EDI
- M3 Extraction (qui correspond à la feuille Excel obtenue après l'export BO)
- Prices (la liste des produits avec leur prix)
- USER (Interface utilisateur)

*Image 1 : Feuille "USER"*

YEAR :		
SEASON :		
CURRENCY :	EUR	
VAT :		
SEASON CODE :		
Methode :	MODEL	
COUNTRY :		
GLN ID :		
STEPS	Instructions	
1	OUVRIR le fichier PRICAT SUPPORT dispo sur le SharePoint	
2	Importer l'extraction M3 issu de BO dans "M3 Extraction"	
3	Completer les éléments de cette fiche (YEAR, SEASON, CURRENCY, VAT, METHODE (voir étape 2), GLN(voir customer DB), COUNTRY (pour le nom du PRICAT, exemple FR))	
4	Appuyer sur le bouton "UPDATE M3 Extraction" cela va completer et ajouter des colonnes dans la feuille (n'appuyer qu'une seule fois ! Sinon reprendre depuis étape 2)	
5	Si tous les résultats de la batterie de tests sont à 0 vous pouvez appuyer sur "UPDATE PRICAT template"	
6	Appuyer sur le bouton "GENERER fichier final", vérifier que tout semble ok et enregistrer le fichier sous un nom correct	
BATTERIE DE TESTS		
TITLE	RESULTS	UPDATE
How many N/A in color :		If 0 -> Ok Else besoin d'ajouter couleur à SUPPORT
How many N/A in FEDAS :		If 0 -> Ok Else Demander à Valetina et/ou Ali sur les codes FEDAS manquants ou Emmanuelle Narese pour une nouvelle extraction
How many N/A in Size Range :		If 0 -> Ok Else
How many N/A in Price :		If 0 -> Ok Else Essayer de changer la méthode

On commence par suivre la même procédure que pour un PRICAT à faire de façon manuelle : on reçoit la demande de catalogue, on demande à notre tour la liste produit/prix qu'on ajoutera cette fois à la feuille prévue à cet effet (Prices), on fait une extraction depuis BO qu'on ajoute à la feuille M3 Extraction.

Dès lors l'utilisateur est amené à compléter la feuille "USER" en remplaçant les informations comme la saison, la devise, le code GLN (pour envoi EDI). L'utilisateur doit ensuite appuyer sur le bouton "UPDATE M3 EXTRACTION" ce bouton exécute une macro VBA qui va créer des colonnes et effectuer les RECHERCHEX entre l'extraction M3 et les feuilles de données (SUPPORT, PRICES). Une batterie de tests s'effectuera pour voir s'il y a des valeurs manquantes et si oui ou sont-elles et quelle est la procédure à suivre pour ne plus les avoir.

Enfin si tous les tests sont à 0, l'utilisateur pour lancer la macro "UPDATE PRICAT TEMPLATE" permettant de remplir l'intégralité de la Template PRICAT, il pourra vérifier que l'exécution s'est bien déroulée (si les dates correspondent bien à la saison, si le code GLN est bien rentré etc ...). Pour finir exécution d'une dernière macro permettant d'exporter le PRICAT complété dans un fichier Excel vierge (sans les feuilles d'extraction et de prix).

Je ne suis pas totalement satisfait du programme mais il permet néanmoins d'accélérer la production d'un PRICAT. Créer cet outil a été plus compliqué que les autres en tout point, d'abord mes connaissances en VBA ne sont pas aussi bonnes que sur d'autres langages même si je n'ai pas eu de mal à trouver les solutions à mes problèmes sur Internet.

Avant mon départ j'ai formé la personne qui sera en charge des PRICAT, tout s'est bien passé dans les explications et la compréhension globale, maintenant ce sujet reste compliqué à appréhender même pour quelqu'un qui a été dessus pendant trois mois comme moi et des développements 100% EDI devraient voir le jour plus tard.

## 2.3. Outils Planification

Voyant ma capacité à développer des outils sur la codification et le PRICAT, j'ai été mis en support sur les outils de planification (ITP) au côté de Quentin PAYEN (extérieur chez Hardis). J'ai notamment eu l'occasion d'effectuer des requêtes SQL assez grande et de découvrir PowerQuery.

Les outils de planifications sont utilisés par les planners, il y en a 2 types :

- Accès manufacture (ce qui est produit par Rossignol)
- Accès négocie (achat au fournisseurs ...)

Les ventes donnent des prévisions, plan supply afin de définir les besoins d'achats et de productions après déduction des stocks.

L'un des outils les plus utilisé est le planning multi article

### 2.3.1. ITP004

Ma première mission sur les outils de planification était de modifier "ITP004\_PLANNING\_MULTI\_ARTICLE". Le PLANNING MULTI ARTICLE est un outil qui permet d'identifier les nouvelles commandes à planifier, de repérer les articles avec messages à traiter, et donne le stock projeté par SKU. Nous avons d'abord dû créer la "DESIRED PLAN DATE" (date à laquelle les produits doivent être disponible au dépôt de Saint-Étienne-de-Saint-Geoirs (SESG)) cette date est égale à la date planifiée (PLAN\_DATE) moins l'écart entre la REQUESTED\_DATE et la CONFIRMED\_DATE, il fallait aussi prendre en compte que si la "DESIRED\_PLANNED\_DATE" tombait un samedi ou un dimanche il fallait la reporter au vendredi. Nous avons ajouté cette nouvelle information avec PowerQuery.

Les commandes ont des motifs comme T10 (à planifier) ou T20 (planifié), ainsi si une ligne respecte certaines conditions on peut lui attribuer l'action "T20 auto".

Plus tard on m'a sollicité pour ajouter une condition à cette action, auparavant si un SKU avait un stock projeté négatif cela n'était pas un problème pour le "T20 auto" mais cela n'était pas cohérent pour les planners, j'ai alors dû ajouter la condition suivante :

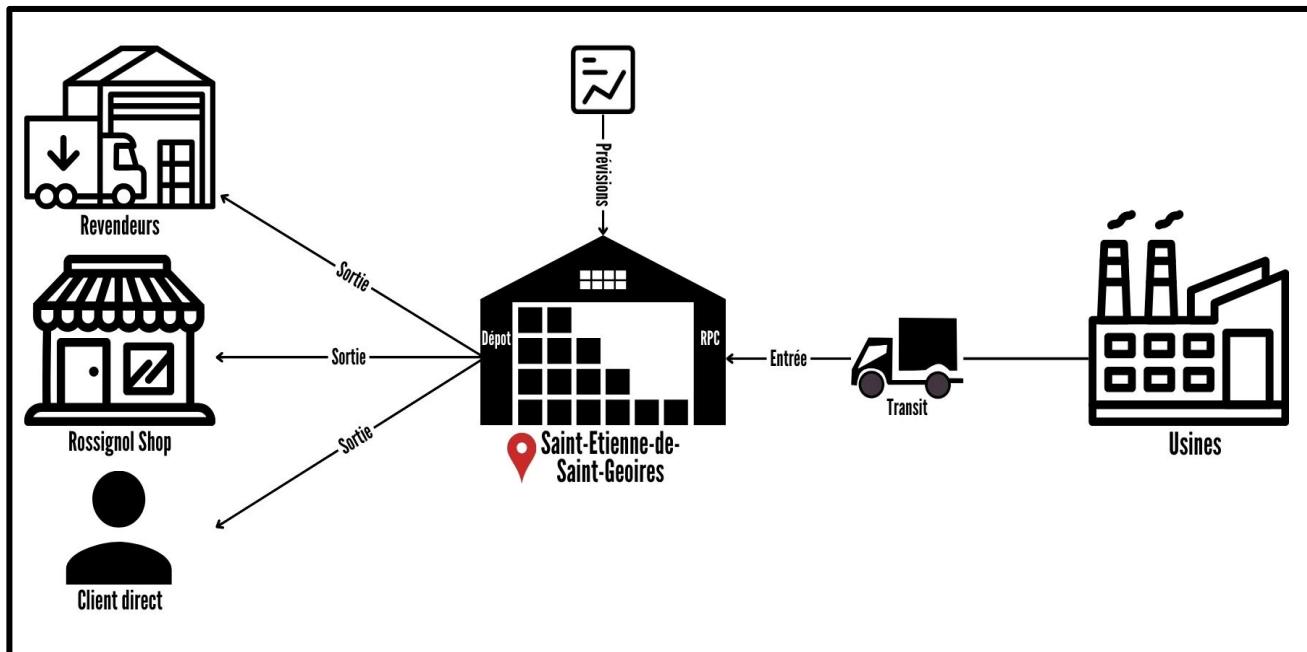
"Si à un moment un SKU a un stock projeté négatif, alors toute les lignes impliquant cet SKU ne peuvent pas être en T20 auto"

Pour mener à bien cette tâche j'ai utilisé les lignes PowerQuery de Quentin qui étaient utilisé pour le même genre de filtre pour les adapter à mon problème, à savoir : identification des lignes impliquant un STDK négatif, regroupement afin d'obtenir une donnée chiffrée, puis fusion afin d'avoir pour toute les lignes d'un SKU les mêmes informations. Enfin j'ai ajouté "ExisteSTDKNegatif >= 0" à la condition "T20 auto".

### 2.3.2. ITP013

Le deuxième outil lui partait plus ou moins de 0, c'est "ITP013\_WEEKLY\_PROJECTED\_PLANNED\_DATE", ce programme a pour but de visualiser la demande des articles ainsi que les prévisions en semaine afin de gérer les stocks et l'approvisionnement nécessaire. Pour cela nous avons effectué une requête SQL sur M3 permettant de récupérer les données nécessaires comme le stock actuel, la marchandise en transit, les commandes prévues ainsi que les prévisions de sorties.

*Schéma 6: Principales entrées et sorties de la plateforme centrale de Rossignol*



Cette fois-ci nous avons intégré la "DESIRED\_PLAN\_DATE" directement dans la requête SQL, car cela permet à Excel de faire moins de calculs derrière. Formatage de la date pour avoir année et semaine et prise en charge des "BACKLOG" (si la semaine est inférieure à la semaine actuelle). Enfin Quentin a réutilisé son code qui provenait d'un autre outil pour effectuer un pivot en PowerQuery ce qui a permis de créer une colonne pour chaque semaine.

Voici un aperçu du résultat final :

*Tableau 5 : Extrait d'une ligne de résultat du programme "ITP013"*

ID PRODUIT	NAME	STOCK	TRANSIT	BACKLOG	202420	...
RANPS02 000169	NOVA 10 XPRESS	279	59	6	4	...

On peut ainsi voir pour chaque article son stock, le nombre d'unité en transit, quantités demandées dans le passé, et quantités demandées sur chaque semaine.

### 2.3.3. ITP015

L'outil "Entry Exit SESG" (ITP015) est conçu pour pouvoir visualiser les entrées et sorties des articles par ligne de produit et par semaine à la plateforme de SESG. Pour cela j'ai créé un outil Excel avec PowerQuery, premièrement l'utilisateur est amené à sélectionner les informations qu'il veut obtenir :

- Entrée ou Sortie
- HardGoods, SoftGoods, Bike

Ensuite une requête SQL est effectuée afin d'extraire toutes les informations en lien avec les filtres rentrés par l'utilisateur, les données sont ensuite regroupées par année, semaine, et ligne de produit. Voici un aperçu du résultat final pour les entrées du HG :

Tableau 6 : Extrait des résultats de "Entry\_Exit\_SESG"

PRODUCT LINE	WEEK	YEAR	QUANTITIES
A1	BACKLOG	BACKLOG	168
A1	22	2024	5610
...	...	...	...
R1	40	2024	2

### 2.3.4. ITP014 & ITP016

ITP014 (DEMAND\_BY\_CURRENT\_PLAN\_DATE) est une copie de ITP013 mais où il n'y a pas de notion de "Desired Plan Date", on remplace cette dernière par la "Plan Date" seule qui elle est une donnée provenant de l'ERP et n'est donc pas à calculer.

ITP016 (SUPPLY\_BY\_CURRENT\_PLAN\_DATE) au contraire de ITP013/14 est créé pour visualiser les entrées, pour cela il a fallu modifier le filtre sur le champ "MOORCA" qui était utilisé pour filtrer sur les sorties + prévisions + transit, on modifie alors pour avoir seulement les entrées et le transit.

Pour ces deux programmes, les résultats sont sous la même forme que le [TABLEAU 5](#)

### 2.3.5. ITP017

Enfin le dernier outil créé est un outil mixant ITP013, ITP014, et ITP016. L'utilisateur a la possibilité de choisir dans Excel quelles données vont être affichées :

- DEMAND\_BY\_DESIRED\_PLAN\_DATE (ITP013)
- DEMAND\_BY\_CURRENT\_PLAN\_DATE (ITP014)
- SUPPLY\_BY\_CURRENT\_PLAN\_DATE (ITP016)

Une division devra être choisie (HG, SG, BK) et des filtres facultatifs sont à sa disposition sur les champs "Ligne Produit", "Chaine SPE2", "Marketing program". Ces champs peuvent être laissé vide, avoir une ou plusieurs valeurs, le programme est adapté aux trois cas.

Selon les choix fait par l'utilisateur on obtient un résultat équivalent au [TABLEAU 5](#)

### 2.3.6. Bilan ITP

Toucher au développement de ces outils m'a permis premièrement d'utiliser mes connaissances en SQL au sein d'une entreprise et de découvrir des nouvelles méthodes comme COALESCE(X, Y) ([CODE 9](#)) qui permet de prendre la première valeur non vide (si X est vide, on prend Y (si il n'est pas vide)), j'ai notamment utilisé cette méthode pour dire que s'il n'y avait pas de DESIRED on prenait la PLAN\_DATE. Deuxièmement j'ai pu découvrir avec Quentin comment utiliser PowerQuery Excel dans un contexte professionnel, en plus d'apprendre de nouveaux outils cela m'a aussi donné l'opportunité de travailler en équipe.

*Code 9 : Utilisation de "COALESCE" dans une requête SQL*

```
CASE
    WHEN DAYOFWEEK(desired_plan_date) = 7 THEN
        desired_plan_date - 1 DAY
    WHEN DAYOFWEEK(desired_plan_date) = 1 THEN
        desired_plan_date
    ELSE
        COALESCE(desired_plan_date, plan_date)
END AS final_plan_date
```

ITP017 est la culmination de mes découvertes entre SQL et PowerQuery car j'ai pu produire un programme prenant en compte les filtres des utilisateurs, en créant des fonctions qui récupérer ces données et en adaptant la requête SQL à ces filtres.

## 3. Conclusion

### 3.1. Intérêt méthodologique et pratique

Mon stage en tant que **gestionnaire de bases de données** chez SKIS ROSSIGNOL a permis de répondre efficacement aux objectifs de départ. L'intérêt méthodologique s'est principalement manifesté par l'optimisation des processus manuels via l'automatisation. En analysant et en révisant les procédures existantes pour la codification des articles dans l'ERP M3 et la préparation des PRICAT, j'ai pu développer des outils facilitant ces tâches, rendant ainsi le travail plus efficient et moins sujet aux erreurs. À titre d'exemple, et selon les retours des utilisateurs, le programme de vérification de correspondance a permis d'économiser des jours voire des semaines de travail.

L'utilisation de Python pour transformer les Line Lists et effectuer des vérifications de correspondance a montré l'importance de la programmation dans le traitement de données complexes. De plus, l'implémentation d'outils en VBA pour semi-automatiser certaines tâches a également prouvé l'efficacité de ces méthodes pour améliorer les opérations quotidiennes. Au total, ce sont près d'un millier de lignes de code Python qui ont été écrites, et près de 500 lignes en VBA/PowerQuery (dont SQL).

Enfin une partie importante de mon travail a aussi consisté à soutenir le développement d'outils de planification, en particulier les outils ITP004, ITP013, ITP014 et ITP015. Ces outils ont permis de mieux gérer les dates de planification et de projection des articles, en utilisant SQL et PowerQuery pour optimiser les processus et les rendre plus robustes.

### 3.2. Bilan personnel

Sur le plan personnel, ce stage a été une expérience enrichissante. Il m'a permis de développer mes compétences techniques en programmation et en gestion de bases de données, tout en m'offrant l'opportunité de travailler dans un environnement dynamique et de collaborer avec des professionnels expérimentés. J'ai appris à mieux comprendre les besoins des utilisateurs finaux et à adapter mes solutions en conséquence. Cette expérience m'a également aidé à renforcer mes capacités de résolution de problèmes et à améliorer ma gestion du temps, des compétences qui seront vraisemblablement utiles le futur.

En conclusion, mon stage chez SKIS ROSSIGNOL a non seulement contribué à améliorer l'efficacité des opérations de l'entreprise mais m'a également permis de grandir en tant que professionnel. Je suis convaincu que les compétences acquises et les leçons tirées de cette expérience me seront extrêmement bénéfiques dans mes projets futurs.

# Annexes

## Document 1 : Guide Utilisateur "SKU\_FROM\_LL.py"

Process\_SKU\_FROM\_LL.md

2024-05-22

## Installation et Utilisation du Programme SKU\_FROM\_LL.py

Le programme **SKU\_FROM\_LL.py** est conçu pour créer un fichier Excel avec l'ensemble des SKU d'une Line List

### Installation

1. Suivez la procédure d'installation d'Anaconda en vous référant au fichier suivant et en ignorant la partie concernant Jupyter. **Je vous recommande de cocher** "Register Anaconda3 as my default Python 3.11" lors de l'installation d'Anaconda.

```
S:\PLA-Planif\Outils_Python\Global Process\Guide d'installation Anaconda et code python.docx
```

2. A la fin de l'installation de Anaconda, lancer **Anaconda Powershell Prompt** (Taper Anaconda dans la barre de recherche Windows et ce dernier devrait apparaître)
3. Une fois dans le répertoire du programme, exécutez la commande suivante pour installer les dépendances nécessaires :

```
pip install pandas
```

### Utilisation

- Dans l'interface de l'invite de commandes Anaconda (**Anaconda Powershell Prompt**), naviguez jusqu'au répertoire contenant le programme en utilisant la commande

```
cd S:\PLA-Planif\Outils_Python\SKU_FROM_LL
```

- Pour utiliser le programme, exécutez la commande suivante dans Anaconda Powershell Prompt depuis le répertoire du fichier (vous pouvez commencer par taper "python S" puis appuyer sur TAB, cela devrait compléter la commande):

```
python ./SKU_FROM_LL_V2.py
```

(Possible que ce ne soit plus V2 à la fin)

- Un explorateur de fichiers s'ouvrira, vous permettant de sélectionner une line list (LL) . Assurez-vous que le fichier ne contient **QUE** la feuille de la LL.

- Si le programme s'exécute sans erreurs, le programme vous demandera alors de rentrer le nom du fichier que vous souhaitez attribuer au fichier Excel.
- Le fichier sera alors enregistrer dans votre onglet 'Téléchargements'

# Guide Développeur "SKU\_FROM\_LL.py"

## Introduction

Le programme "SKU\_FROM\_LL.py" a pour objectif de créer une Line List sous la forme SKU ainsi que les Samples

## Organisation

Structure :



Développement :

Le programme utilise 2 fonctions (qui ont toutes une docstrings):

- `create_model_color()`

Création du modèle couleur en fonction de la longueur du modèle et de la couleur

- `chaine_en_liste()`

Transforme une chaîne de caractère avec des caractères de délimitation (comme '-') en liste

Le programme peut être découpé en 3 parties distinctes :

- Récupération des données issue de la LL (fichiers Excel)
- Traitement : création de colonnes, explosion des tailles pour créer les SKU
- Création des Samples
- Export des Dataframe -> Création de feuilles Excel pour les SKU et les Samples

# Installation et Utilisation du Programme check\_M3\_LL.py

Le programme **check\_M3\_LL.py** est conçu pour vérifier si les éléments du PLM/KUBIX sont correctement entrés dans M3.

## Structure de l'outil

- Le dossier *DATA* dans *Other* peut vous servir à mettre les fichiers de LL à vérifier, vous pouvez créer des dossiers pour chaque saison
- Le dossier *MAPPING\_MDI* dans *Other* contient tout les mappings MDI dont vous pourriez avoir besoin pour effectuer des modification par MDI
- Le dossier *Project\_M3\_LL* est le dossier principal de l'outil, il regroupe les scripts python, le fichier *requirements.txt* et un dossier *MAPPING\_MDI* qui contient les "templates" MDI relative à l'outil

## Installation

1. Suivez la procédure pour créer un lien ODBC vers la base M3 en vous référant au guide d'installation suivant qui permettra de se connecter à la base M3 et d'effectuer des requêtes.

S:\PLA-Planif\Outils\_Python\Global Process\Guide d'installation d'un lien ODBC vers une base de données M3.docx

2. Suivez la procédure d'installation d'Anaconda en vous référant au fichier suivant et en ignorant la partie concernant Jupyter. **Je vous recommande de cocher "Register Anaconda3 as my default Python 3.11"** lors de l'installation d'Anaconda.

S:\PLA-Planif\Outils\_Python\Global Process\Guide d'installation Anaconda et code python.docx

3. A la fin de l'installation de Anaconda, lancer **Anaconda Powershell Prompt** (Taper Anaconda dans la barre de recherche Windows et ce dernier devrait apparaître)
4. Dans l'interface de l'invite de commandes Anaconda, naviguez jusqu'au répertoire contenant le programme en utilisant la commande

```
cd S:\PLA-Planif\Outils_Python\CHECK_M3_LL
```

5. Une fois dans le répertoire du programme, exécutez la commande suivante pour installer les dépendances nécessaires :

```
pip install -r ./requirements.txt
```

## Utilisation

- Naviguer jusqu'au répertoire contenant le fichier python en executant cette commande dans **Anaconda Powershell Prompt** :

```
cd S:\PLA-Planif\Outils_Python\CHECK_M3_LL\main
```

- Pour utiliser le programme, exécutez la commande suivante depuis le répertoire du fichier (vous pouvez commencer par taper "python c" puis appuyer sur TAB, cela devrait compléter la commande):

```
python ./check_M3_LL_V2-4.py
```

(Possible que ce ne soit plus V2-4 mais une version supérieure)

- Un explorateur de fichiers s'ouvrira, vous permettant de sélectionner les line lists (LL) que vous souhaitez vérifier (que vous avez mis ou non dans *DATA (LL)*). Assurez-vous que les LL sélectionnées sont de la même saison et qu'elles ne contiennent que la feuille de la LL.
- Le programme vous demandera ensuite de saisir la saison (par exemple "26FW"), ce qui permettra à la requête de récupérer tous les articles entre la saison entrée - 2 saisons (par exemple, 25FW) et la dernière saison.
- Si le programme s'exécute sans erreurs, vous recevrez un message indiquant que le fichier a été téléchargé dans votre répertoire de téléchargement. Si aucune erreur n'est détectée, vous recevrez également un message.
- Le dossier résultant contient un résumé des erreurs dans **M3\_Error\_Report** et les fichiers CSV utiles à la modification par MDI s'il y a un nombre important d'erreurs sur le genre, l'origine, la saison, etc.
- Consultez le fichier Excel **M3\_Error\_Report** pour identifier les erreurs. Il comporte deux feuilles : une pour les erreurs sur les modèles (MODEL) et l'autre pour les erreurs au niveau SKU. Chaque ligne comprend les informations de base (Code MODEL/SKU, nom, et de quelle LL il provient) et les colonnes "PROBLEM" où les erreurs sont répertoriées.
- Pour utiliser les fichiers MDI, référez-vous aux mappings disponibles dans le dossier : *MAPPING\_MDI*

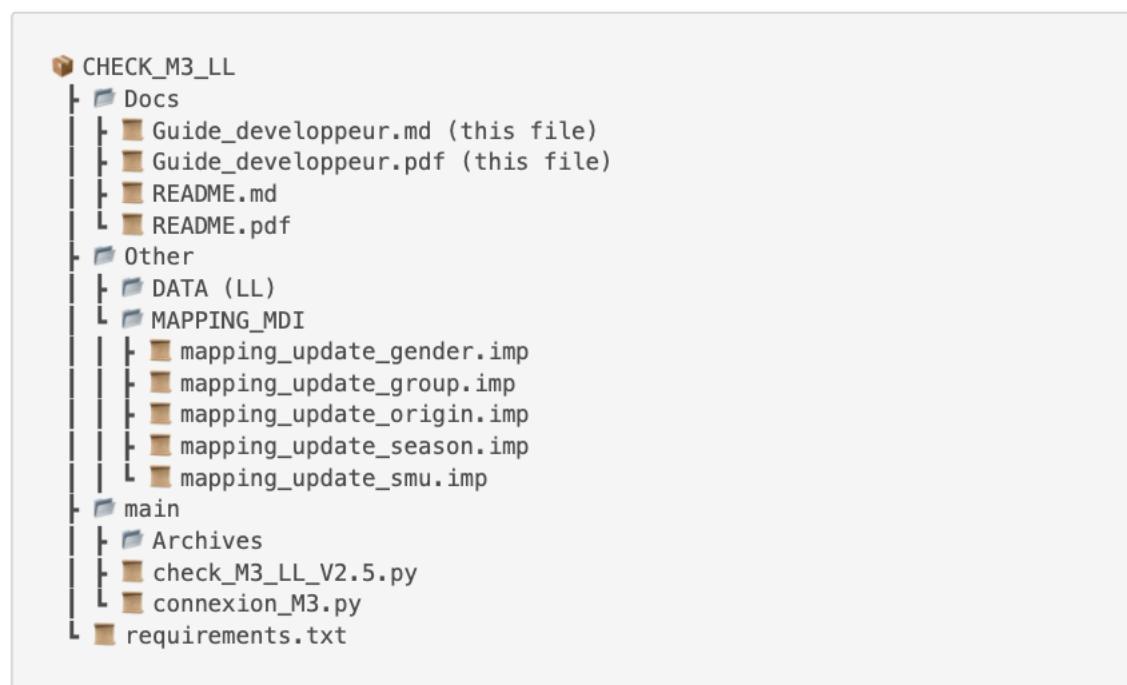
# Guide Développeur "check\_M3\_LL.py"

## Introduction

Le programme "check\_M3\_LL.py" a pour objectif de vérifier les correspondances entre les Line List et l'ERP M3.

## Organisation

Structure :



Développement :

Le programme utilise 6 fonctions (qui ont toutes une docstrings):

- **connect\_to\_database()**  
Permet de se connecter à la base M3 grâce à la connexion ODBC.
- **create\_model\_color()**  
Création du modèle couleur en fonction de la longueur du modèle et de la couleur
- **chaine\_en\_liste()**  
Transforme une chaîne de caractère avec des caractères de délimitation (comme '-') en liste
- **create\_seasons\_range()**  
Création d'un tuple qui récupère toutes les saisons comprises dans une liste à partir d'une saison donnée (utilisation du tuple dans la requête SQL)

- **extract\_brand\_code()**

Extraction du code de la marque qui est entre paranthèse dans la LL

- **create\_mdi\_mms001()**

Création d'un DataFrame compatible avec les MDI de la base article (MMS001)

Le programme peut être découpé en 5 parties distinctes :

- Extraction des données avec la requête SQL sur M3 et récupération des données issues des LL (fichiers Excel)
- Transformation des données afin d'avoir un format où les 2 DataFrames peuvent communiquer
- Comparaison des informations entre M3 et LL -> Identification des erreurs
- Découpage des erreurs au niveau modèle et au niveau SKU
- Retranscription des problèmes -> Création de feuilles Excel pour un résumé et de fichier CSV pour modification par MDI

### *Code 10 : fonction "create\_model\_color"*

```
● ● ●
```

```
def create_model_color(model):
    """Créer le modèle couleur en identifiant combien de caractère fais le
    code modèle

    Args:
        model (str): valeur du code modèle

    Returns:
        Modèle couleur pour chaque ligne du dataframe
    """
    if len(model['product code']) == 7:
        return model['product code'] + ' ' + model['color'][3:]
    elif len(model['product code']) == 8:
        return model['product code'] + ' ' + model['color'][3:]
    else:
        return None
```

### *Code 11 : fonction "chaine\_en\_liste"*

```
● ● ●
```

```
def chaine_en_liste(chaine, split_chars):
    """Transforme la chaîne des tailles pour créer une liste des tailles

    Args:
        chaine (str): la chaîne des tailles (exemple : 00M-00S)
        split_chars (list): les caractères sur lesquelles on coupe (ici -)

    Returns:
        list: liste des tailles (exemple : [00M, 00S])
    """
    if isinstance(chaine, str):
        chaine = chaine.replace(' ', '')
        for char in split_chars:
            chaine = chaine.replace(char, '-') # Remplacer chaque caractère
        de split_chars par un tiret
        return chaine.split('-') # Diviser la chaîne résultante par les
        espaces
    else:
        return [str(chaine)]
```

### *Code 12 : fonction "create\_seasons\_range"*



```
def create_seasons_range(season: str, seasons: list):
    """Créer une plage de valeurs possible dans laquelle la requête SQL
    devra chercher

    Args:
        season (str): une saison choisi par l'utilisateur (exemple : 25SS)
        seasons (list): la liste de toute les saisons

    Returns:
        tuple: un tuple de la saison sélectionner puis celles qui viendront
    (exemple : 25SS->35SS)
    """
    # Obtenir l'index de la saison choisie dans la liste
    index_season = seasons.index(season)

    # Créer un tuple avec la saison choisie et les saisons suivantes jusqu'à
    la fin de la liste
    seasons_range = tuple(seasons[(index_season-2):])
    return seasons_range
```

### *Code 13 : fonction "extract\_brand\_code"*



```
def extract_brand_code(brand: str):
    """Extrait la marque du modèle

    Args:
        brand (str): l'intitulé de la marque

    Returns:
        str: le code de la marque (ROSS pour ROSSIGNOL)
    """
    match = re.search(r'\((.*?)\)', brand)
    if match:
        return match.group(1)
    else:
        return brand
```

#### Code 14 : Fonction "create\_mdi\_mms001"

```
def create_mdi_mms001(problem_field: str, mdi_field: str,
dico_correspondance=None):
    """Créer le dataframe pour un MDI MMS001

    Args:
        problem_field (str): le nom de la colonne du dataframe
        'df_model_problem' et 'df_SKU_problem' (exemple : 'GENDER PROBLEM')
        mdi_field (str): Le nom du champ dans M3/MDI correspondant au
        problème (exemple : CFI5 pour GENDER)
        dico_correspondance (dict, optional): Si il est nécessaire un
        dictionnaire de correspondance. Par défaut à None.

    Returns:
        Dataframe complété prêt à être exporter en CSV
    """

    df_field_model = df_model_problem.dropna(subset=problem_field)
    df_field_model = df_field_model[df_field_model[problem_field] != '']

    df_field_sku = df_SKU_problem.dropna(subset=problem_field)
    df_field_sku = df_field_sku[df_field_sku[problem_field] != '']

    df_field_problem = pd.concat([df_field_sku, df_field_model])

    mdi_maj = pd.DataFrame(columns=['CONO', 'ITNO', mdi_field])

    for _, row_fp in df_field_problem.iterrows():

        # Obtenir la valeur de 'problem field' après 'LL :
        field = row_fp[problem_field].split('LL : ')[1].split(',')[0]

        if isinstance(row_fp['SKU'], list):
            # Si c'est une liste, concaténer avec la colonne 'MODEL'
            regroup = [row_fp['MODEL']] + row_fp['SKU']
        else:
            # Si ce n'est pas une liste, créer une liste avec la valeur de
            # la colonne 'SKU' et concaténer avec la colonne 'MODEL'
            regroup = [row_fp['MODEL'], row_fp['SKU']]

        # Itérer sur les SKU
        for item_id in regroup:
            # Créer un dictionnaire temporaire pour chaque SKU
            temp_mdi = {'CONO': 300, 'ITNO': item_id, mdi_field: field}

            # Ajouter le dictionnaire temporaire à mdi_maj_gender
            mdi_maj.loc[len(mdi_maj)] = temp_mdi

        if dico_correspondance:
            dico_correspondance_inverse = {v: k for k, v in
dico_correspondance.items()}
            mdi_maj[mdi_field] =
mdi_maj[mdi_field].replace(dico_correspondance_inverse)

    return mdi_maj
```