

Université de Montpellier
FACULTÉ DES SCIENCES DE MONTPELLIER
M1 IASD
Projet de TER

**ANALYSE, TRAITEMENT ET INTÉGRATION DE
DONNÉES CLINIQUES POUR APPLICATIONS
DANS LE DOMAINE HOSPITALIER**



Rédigé par :

Bachir HADJOUDJA 21811363
Rania BENDAHMANE 21811387
Rym ZEGGAR 21909615
Yousef LABIAD 21710780

Encadrants :

Djamel SERIAI
Billel SERIAI

Table des matières

1	Introduction	2
1.1	Contexte et problème	2
1.2	Objectifs	2
2	Étude des concepts, des technologies et des outils	3
2.1	Outils/technologies utilisés	3
2.2	Les concepts liés à la standardisation des données dans le domaine hospitalier	4
2.2.1	HL7	4
2.2.2	FHIR	4
i	Description	4
ii	Les composants du FHIR	4
iii	Les différences entre le HL7 v2.X et le FHIR	5
2.2.3	HER	5
i	Qu'est-ce que un EHR	5
ii	Qui peut y accéder ?	6
iii	Informations incluses dans un EHR	6
iv	Prinicepe	7
2.3	Analyse des besoins et compréhension des données cliniques	7
2.3.1	Importance et problèmes d'intégration de données	7
2.3.2	Enjeux spécifiques dans le domaine de la santé	8
2.4	Le processus ETL	8
2.4.1	Les 3 fonctions fondamentales d'un ETL	9
2.4.2	Les différents composants d'un système ETL	9
2.4.3	Exemples d'usage	10
2.5	Étude des outils	11
2.5.1	Trifacta	11
i	C'est quoi trifacta ?	11
ii	Comment on peut s'en servir de Trifacta dans le domaine médical ?	11
iii	Exemple d'utilisation	12
iv	Résultats finaux	13
2.5.2	OpenRefine	14
2.5.3	Étude de NiFi	16
i	Présentation de l'outil	16
ii	Utilisation de Nifi dans un contexte hospitalier	17
iii	Les différents processeurs nifi	17
iv	Fichier de configuration Nifi Propreties	21
2.5.4	Comparaison et choix des outils	22

3 Approche méthodologique	23
3.1 Cahier de charges	23
3.2 Conception du projet	25
3.2.1 Architecture du système	25
3.2.2 Sélection des outils	25
3.2.3 Processus ETL	25
3.2.4 Interface utilisateur	25
3.2.5 Diagramme de l'architecture du système	25
4 Réalisation et résultats	26
4.1 Mise en oeuvre du processus ETL pour le traitement de données cliniques	26
4.1.1 Avant de commencer : Anonymisation des données	26
i Contexte	26
ii Méthodologie	27
4.1.2 Première étape : "EXTRACT"	28
i Fichier patient	28
ii Fichier pharmacie	29
iii Fichier radiologie	30
iv Fichier transfert	31
v Fichier Laboratoire	32
vi Fichier Diagnosis	33
4.1.3 Deuxième étape : "TRANSFORM"	34
i Process Group	35
ii Prétraitements	38
iii Règles d'intégration	39
4.1.4 Troisième étape : "LOAD"	45
4.2 IHM et architecture logicielle pour l'interaction avec les composants du processus ETL	50
4.2.1 Conception de l'interface	50
4.2.2 Implémentation de l'interface	51
4.2.3 Explication détaillée de l'interface graphique	54
4.2.4 Utilisation de Docker pour la création d'une image NiFi	60
i Vue détaillée sur Docker	60
ii Utilisation de Docker	61
4.3 Bilan	63
4.4 Difficultés rencontrées	63
4.5 Solutions apportées	63

5 Gestion du projet	64
5.1 Organisation	64
5.1.1 Description de l'organisation	64
i théorie	64
ii Pratique	64
5.1.2 Outils Gestion de Projet	64
5.1.3 Outils de recherche	65
5.2 Diagramme de Gantt	66
6 Conclusion	66
6.1 Récapitulatif des travaux réalisés	66
6.2 Perspectives d'amélioration	67
7 Bibliographie	68

Remerciements

Nous remercions tout particulièrement M. Djamel Seriai, notre encadrant, pour son précieux encadrement, son soutien constant et ses conseils avisés tout au long de la réalisation de ce projet. Sa grande expertise et sa disponibilité ont été d'une grande valeur pour nous.

Nous tenons également à exprimer notre gratitude envers M. Billel Seriai, data analyst à l'entreprise Power Health, pour son assistance technique et ses conseils éclairés tout au long du développement de ce projet. Sa contribution a été essentielle pour la réussite de notre travail.

Nous voulons également exprimer nos sincères remerciements à Madame Madalina Croitoru et aux intervenants de son cours pour nous avoir aidés à y voir plus clair dans notre organisation du projet.

1 Introduction

1.1 Contexte et problème

Au sein de notre rapport, nous rendons compte des activités réalisées dans le cadre du projet intitulé "Automatisation de l'analyse, du traitement et de l'intégration de données cliniques dans le domaine hospitalier", réalisé en partenariat avec l'entreprise PowerHealth.

L'objectif principal de ce projet était de mettre en place un processus automatisé permettant d'analyser, traiter et intégrer des données cliniques provenant de différents logiciels utilisés dans le domaine médical. Cela, dans un cadre expérimental car, en effet, il nous a été demandé de comparer divers outils en vue de déterminer lequel était le plus adapté à ce genre de travail. Ces expérimentations nous ont permis d'évaluer les fonctionnalités, les performances et les avantages de chaque outil, en prenant en compte les exigences spécifiques du projet. Une fois les expérimentations terminées, nous avons sélectionné l'outil qui répondait le mieux à nos besoins et nous avons utilisé celui-ci pour fournir le travail final.

De nos jours, l'ère du numérique est marquée par une explosion de la quantité de données générées et recueillies dans divers domaines, notamment dans le secteur de la santé. Ces données, en raison de leur volume, de leur variété et de leur vitesse, représentent un trésor d'informations potentiellement précieuses pour la recherche médicale, l'amélioration de la qualité des soins, et la mise en place de politiques de santé publique efficaces.

Cependant, l'utilisation efficace de ces données présente des défis significatifs, en particulier dans le domaine médical et hospitalier, où la gestion des données cliniques constitue un défi majeur en raison de la diversité des systèmes d'information et des logiciels utilisés.

Les données cliniques sont souvent générées et stockées dans des formats différents, dans des applications spécialisées et des bases de données décentralisées. Cette fragmentation des données crée des obstacles à leur utilisation efficace et à leur exploitation pour améliorer les soins des patients.

1.2 Objectifs

L'homogénéisation des données médicales et l'interopérabilité sont des concepts clés pour résoudre ce problème. L'homogénéisation vise à standardiser les données cliniques, en les structurant de manière cohérente et en harmonisant les terminologies utilisées. Cela permet d'assurer la comparabilité et la cohérence des données entre différents systèmes et établissements de santé. L'interopérabilité, quant à elle, concerne la capacité des systèmes informatiques à échanger, à interpréter et à utiliser les données de manière transparente et fluide.

L'enjeu est de transformer ces données disparates en un format standardisé et cohérent, facilitant leur intégration et leur utilisation ultérieure. Pour cela, nous utilisons des outils et des techniques de traitement des données pour extraire, nettoyer, normaliser et harmoniser les informations cliniques. L'automatisation de ce processus permet de réduire les erreurs et les tâches manuelles fastidieuses, tout en garantissant la qualité et l'intégrité des données. Une fois les données homogénéisées, l'interopérabilité devient possible. Les informations cliniques peuvent être partagées et échangées plus facilement entre les différents acteurs du domaine médical, tels que les professionnels de la santé, les hôpitaux, les laboratoires et les assureurs. Cela favorise une meilleure coordination des soins, une prise de décision plus éclairée et une optimisation des processus cliniques.

Ce rapport détaille les différentes étapes du projet, depuis la collecte des données cliniques jusqu'à la mise en œuvre du processus d'automatisation et du développement de l'interface. Nous présentons également les défis techniques rencontrés et les solutions mises en place pour les surmonter. De plus, nous mettons en évidence les compétences logicielles acquises tout au long du projet, telles que l'utilisation de logiciels spécifiques au domaine médical et le développement d'interfaces web interactives.

2 Étude des concepts, des technologies et des outils

2.1 Outils/technologies utilisés

Dans cette section, nous aborderons les outils métier et les technologies utilisés pour mener à bien notre projet d'intégration de données cliniques : Pour mener à bien notre projet, nous avons utilisé diverses technologies, notamment :

- **Apache Nifi** : Une plateforme de gestion des flux de données qui permet d'automatiser et d'orchestrer les processus d'intégration de données entre différentes sources et destinations.
- **OpenRefine** : Un outil d'édition et de nettoyage de données puissant et flexible, qui aide à préparer et à transformer les données avant de les intégrer dans notre système.
- **Trifacta** : Une plateforme de préparation des données qui permet de nettoyer, structurer et enrichir les données pour faciliter leur analyse et leur intégration.
- **Nipyapi** : Une bibliothèque Python pour interagir avec l'API REST d'Apache Nifi, qui nous a permis d'automatiser et de gérer nos flux de travail ETL. Grâce à Nipyapi, nous avons pu contrôler, surveiller et gérer efficacement les processus d'intégration de données en automatisant certaines tâches, en optimisant les performances et en simplifiant la gestion des flux de données. Cette bibliothèque a facilité la configuration, le déploiement et la maintenance de nos solutions d'intégration de données, garantissant ainsi un processus d'intégration fluide et efficace pour les données cliniques.
- **Python** : Un langage de programmation polyvalent et populaire utilisé pour diverses tâches liées à notre projet, y compris le traitement des données, la création de scripts pour automatiser certaines étapes du processus ETL, et l'analyse des données cliniques.
- **Tkinter** : Une bibliothèque Python pour créer des interfaces graphiques utilisateur (GUI) pour nos applications. Dans le cadre de notre projet, Tkinter a été utilisé pour développer une interface conviviale et interactive, permettant aux utilisateurs de sélectionner et d'appliquer facilement des règles de transformation et de validation sur leurs fichiers sources pour l'intégration de données cliniques.
- **Convertdate** : Une bibliothèque Python qui nous permet de convertir facilement les dates entre différents formats et calendriers.
- **Rest** : Un style d'architecture logicielle qui permet la communication entre systèmes sur Internet. Dans notre projet, l'API REST a été utilisée pour faciliter l'échange de données entre nos applications et les bases de données distantes. Elle permet aux utilisateurs d'envoyer et de recevoir des données de manière efficace et sécurisée, en utilisant les méthodes standard du protocole HTTP comme GET, POST, PUT et DELETE.
- **OpenPyXL** : Une bibliothèque Python open source qui permet de lire et d'écrire des fichiers Excel (XLSX / XLSM / XLTM / XLTX) en utilisant le langage de programmation Python. Cette bibliothèque fournit une interface simple et efficace pour manipuler des données dans des fichiers Excel, y compris la création, la modification et la suppression de feuilles de calcul, de cellules, de lignes et de colonnes. Elle peut également gérer les formules, les graphiques, les images, les commentaires et les styles de cellules. Openpyxl est largement utilisé dans l'automatisation de tâches de traitement de données qui impliquent des fichiers Excel.

- **Docker** : Une plateforme open-source qui permet d’automatiser le déploiement, la gestion et l’exécution d’applications dans des conteneurs. Un conteneur Docker est une unité logicielle légère et autonome qui encapsule une application ainsi que tous les éléments nécessaires à son fonctionnement, tels que les bibliothèques, les dépendances et les fichiers de configuration. Il offre une approche innovante pour la virtualisation au niveau du système d’exploitation (OS-level virtualization), permettant aux applications d’être isolées et portables, tout en utilisant efficacement les ressources système.

L’utilisation de ces outils et technologies a permis à notre équipe de travailler efficacement sur le projet, en assurant une intégration réussie et en garantissant la qualité et la cohérence des données.

2.2 Les concepts liés à la standardisation des données dans le domaine hospitalier

2.2.1 HL7

Health Level Seven International (HL7) est un organisme de normalisation à but non lucratif fondé en 1987. Il se consacre à fournir un ensemble de normes internationales pour le transfert de données cliniques et administratives entre des applications logicielles utilisées par divers prestataires de soins de santé. Les normes HL7 sont utilisées dans le monde entier et jouent un rôle clé dans l’interopérabilité des systèmes d’information sur la santé. Ces normes se concentrent sur la septième au septième niveau (couche application) du modèle de communication de l’Organisation internationale de normalisation (ISO) pour l’interconnexion des systèmes ouverts (OSI). et sont adoptées par d’autres organismes de normalisation tels que l’American National Standards Institute et l’Organisation internationale de normalisation.

2.2.2 FHIR

i Description

Le FHIR correspond à un acronyme pour Fast Healthcare Interoperability Resources. Il s’agit d’un standard de communication de données de santé.

Ce dernier est utilisé en vue de permettre la transmission efficace et sécurisée d’informations de santé entre différents systèmes informatiques.

A l’image du HL7 v2.X, il permet aux différents acteurs du système de santé de partager des informations de manière cohérente et fiable, ce qui peut améliorer la qualité des soins fournis aux patients. Il a un objectif similaire au HL7 v2.X, dans la mesure où il a été développé par HL7 (Health Level Seven International).

ii Les composants du FHIR

Le FHIR se compose de ressources standardisées (qui sont des éléments de données normalisés utilisés pour représenter différentes informations de santé), telles que les patients, les médicaments, les observations de laboratoire, les ordonnances, etc.

Ces ressources standardisées peuvent être utilisées pour représenter les différents éléments de données de santé. Chaque ressource est décrite selon une structure standard, qui définit les différents champs et les types de données associées à cette ressource.

Par ailleurs, le FHIR utilise des API (Application Programming Interfaces) pour permettre la communication entre les différents systèmes informatiques par une possibilité d'interconnexion de ces systèmes.

iii Les différences entre le HL7 v2.X et le FHIR

On est amenés à penser que le HL7 v2.X et le FHIR sont similaires du moment où tous les deux sont des protocoles de communication de données de santé. Cependant, il existe certaines différences clés entre ces deux standards qu'il est intéressant de soulever :

- **Objectif** : HL7 v2.X est un protocole de communication plus ancien et plus large, qui a été créé pour permettre l'échange de données de santé entre les systèmes informatiques. Quant au FHIR, il a été spécifiquement développé pour utiliser de nouvelles technologies, à l'image des API, pour permettre une communication plus efficace et plus flexible des données de santé.
- **Complexité** : HL7 v2.X est un protocole complexe qui peut être difficile à utiliser et à mettre en œuvre. FHIR, en revanche, est conçu pour être plus simple et plus facile à utiliser.
- **Flexibilité** : FHIR est conçu pour être plus flexible et plus facilement adaptable à différents scénarios et utilisations que HL7 v2.X.
- **Utilisation des données** : HL7 v2.X est principalement utilisé pour les transactions de données de base telles que les ordonnances et les résultats de laboratoire. FHIR, d'un autre côté, peut être utilisé pour un plus large éventail de scénarios, notamment les applications mobiles de santé et les systèmes de gestion de la santé à distance.

Voici un tableau récapitulatif :

	HL7	FHIR
Objectif	Échange de données de base entre les systèmes informatiques	Communication plus efficace et flexible des données de santé
Complexité	Protocole complexe	Protocole simple et facile à utiliser
Flexibilité	Moins flexible	Plus flexible et facilement adaptable
Utilisation des données	Transactions de base telles que les ordonnances et les résultats de labo	Applications mobiles de santé et systèmes de gestion de la santé à distance
Développeur	Health Level Seven International	Health Level Seven International

2.2.3 HER

i Qu'est-ce que un EHR

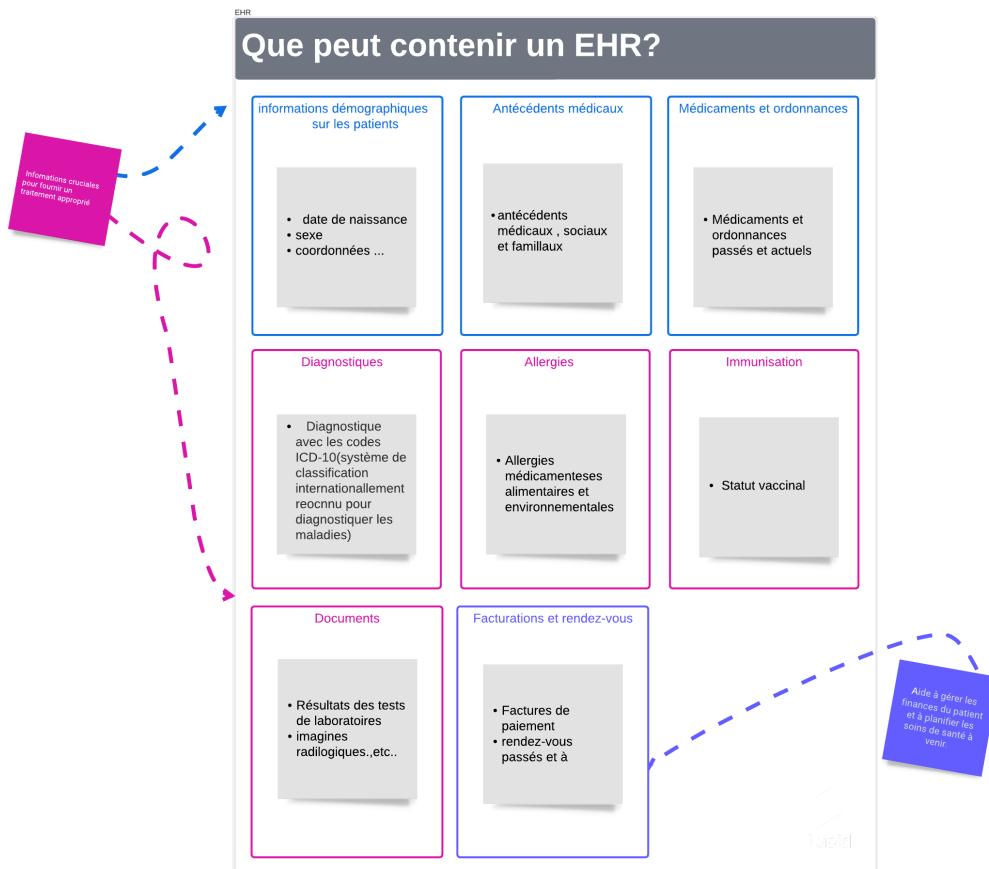
Un dossier de Santé Electronique (EHR) est un document officiel de la santé d'un individu qui peut être partagé entre plusieurs installations et agences. C'est un moyen numérique de stocker les informations de santé d'un patient de manière sécurisée et en temps réel, ce qui les rend accessibles instantanément pour les utilisateurs autorisés. Cela offre une vue complète et numérique de l'historique médical d'un patient, remplaçant ainsi le dossier papier traditionnel.

ii Qui peut y accéder?

Les dossiers de santé électroniques peuvent être partagés entre différents prestataires de soins de santé, tels que les médecins, les hôpitaux, les laboratoires, les centres d'imagerie médicale, les pharmaciens, qui sont impliqués dans les soins d'un patient.

Les EHR permettent une accessibilité instantanée et sécurisée aux informations par les utilisateurs autorisés.

iii Informations incluses dans un EHR



iv Principe

Un exemple de l'utilisation des normes HL7 dans la pratique est celui d'un hôpital où les informations relatives aux patients sont stockées dans un système de dossiers médicaux électroniques (EHR). Lorsqu'un patient est adressé à un spécialiste ou reçoit des soins dans un autre établissement, le nouveau prestataire de soins doit avoir accès aux antécédents médicaux du patient pour garantir un traitement approprié et éclairé. Grâce aux normes HL7, le DME du patient peut être transmis avec précision et en toute sécurité d'un prestataire de soins à un autre, ce qui facilite la prestation de soins transparents et efficaces. Ce processus ne serait pas possible sans le format normalisé d'échange d'informations médicales établi par HL7.

2.3 Analyse des besoins et compréhension des données cliniques

2.3.1 Importance et problèmes d'intégration de données

L'intégration de données présente divers problèmes qui revêtent une importance tant au niveau général que dans le domaine de la santé. En général, l'un des principaux problèmes réside comme expliqué précédemment, dans la diversité des sources de données. Les organisations collectent des informations à partir de multiples systèmes et plates-formes, chacun ayant sa propre structure, format et terminologie. Cela rend l'intégration des données complexe et laborieuse, car il faut harmoniser les différentes structures et aligner les terminologies pour obtenir une vue cohérente et unifiée des informations.

Un autre défi courant est lié à la qualité des données. Les données peuvent contenir des erreurs, des

doublons, des incohérences ou des lacunes, ce qui peut compromettre la fiabilité et l'utilité des informations intégrées. La gestion de la qualité des données est donc cruciale pour garantir la pertinence et la précision des résultats obtenus à partir des données intégrées.

2.3.2 Enjeux spécifiques dans le domaine de la santé

Dans le domaine de la santé, ces problèmes d'intégration de données sont amplifiés. Les établissements de santé génèrent et stockent une quantité massive de données provenant de diverses sources, telles que les dossiers médicaux électroniques, les dispositifs de surveillance, les résultats de laboratoire et les rapports d'imagerie médicale. L'intégration de ces données est essentielle pour obtenir une vue d'ensemble complète du patient, faciliter la coordination des soins et prendre des décisions médicales éclairées.

L'interopérabilité des systèmes d'information en santé est également un enjeu majeur. Les établissements de santé utilisent souvent des logiciels et des systèmes propriétaires qui ne communiquent pas facilement entre eux. Cela limite la capacité des professionnels de la santé à accéder et à échanger des informations vitales, entraînant des retards, des erreurs et une coordination inefficace des soins. L'intégration des données et l'interopérabilité permettent de surmonter ces obstacles en facilitant l'échange fluide des informations entre les différents acteurs de la santé, améliorant ainsi la qualité et l'efficacité des soins.

Dans ce contexte, l'utilisation d'outils ETL (Extract, Transform, Load) peut aider à faciliter l'intégration de données dans le domaine de la santé. Les processus ETL permettent d'extraire des données de différentes sources, de les transformer en un format uniforme et de les charger dans une base de données ou un autre système de stockage pour une utilisation ultérieure.

Grâce à l'intégration de données, les professionnels de la santé ont accès à une vision plus complète et précise des patients, ce qui facilite la prise de décision éclairée et améliore les soins prodigués. L'un des principaux avantages de l'intégration de données est la possibilité de rassembler des informations provenant de différentes sources, telles que les dossiers médicaux électroniques, les résultats de laboratoire, les données de dispositifs médicaux et les données provenant de capteurs. En consolidant ces données disparates, il devient possible de créer un profil complet du patient, permettant aux professionnels de la santé de mieux comprendre son historique médical, d'identifier les risques potentiels et de recommander des traitements personnalisés.

De plus, l'intégration de données facilite la collaboration interdisciplinaire en permettant aux différents acteurs du système de santé d'accéder aux mêmes informations. Les médecins, les infirmières, les spécialistes et les administrateurs peuvent partager des données en temps réel, ce qui améliore la coordination des soins et réduit les erreurs médicales.

L'utilisation de données intégrées dans le domaine de la santé permet également de réaliser des analyses approfondies et de mettre en œuvre des approches basées sur les données. Les algorithmes d'apprentissage automatique et l'intelligence artificielle peuvent être appliqués aux données intégrées pour détecter des schémas, prévoir des tendances et prendre des décisions prédictives. Cela ouvre de nouvelles opportunités pour la recherche médicale, la détection précoce des maladies, l'optimisation des protocoles de traitement et la gestion efficace des ressources.

2.4 Le processus ETL

Les ETL (Extract, Transform, Load) signifient **Extraction**, **Transformation**, **Chargement**, ce sont des processus utilisés pour extraire des données d'une ou plusieurs sources, les transformer pour les adapter à des besoins spécifiques et les charger dans une base de données de destination. Les ETL sont largement utilisés pour centraliser les données dans les entreprises pour une analyse et une

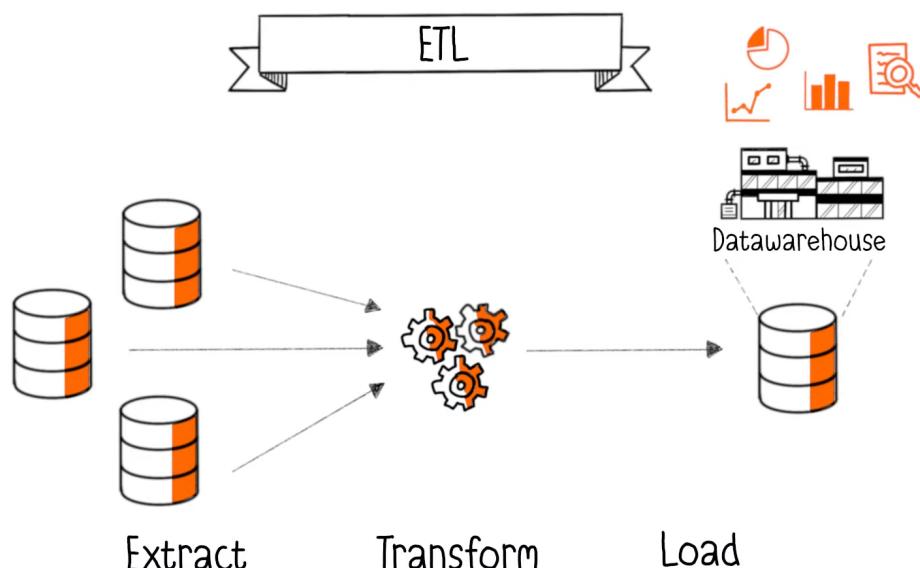
utilisation ultérieure.

L'ETL est souvent le processus spécifique requis pour charger les données vers et depuis les data marts et les entrepôts de données, mais c'est un processus qui est également utilisé pour convertir (transformer) les bases de données d'un format ou d'un type à un autre.

2.4.1 Les 3 fonctions fondamentales d'un ETL

- **Extraction** : processus de collecte de données à partir de différentes sources, telles que les bases de données, les fichiers, les applications en ligne, etc.
- **Transformation** : processus de modification des données collectées pour les adapter à un format ou une structure spécifiques. Cela comprend des opérations telles que la suppression de données redondantes, la fusion de données provenant de sources différentes, la conversion de formats de données, etc.
- **Chargement** : processus de transfert des données transformées vers une base de données ou un autre système de stockage central.

Ensemble, ces fonctions permettent de garantir que les données sont cohérentes, fiables et prêtes à l'utilisation pour une analyse et une exploitation ultérieure.



2.4.2 Les différents composants d'un système ETL

- **Connecteurs** : les connecteurs ETL permettent d'exporter ou d'importer les données dans les applications métiers, et de les rendre lisibles. Des éditeurs tels qu'Oracle ou SAP mettent ainsi à disposition des connecteurs pour rendre utilisables leurs outils avec des bases de données très diverses. Ils sont utiles pour plusieurs raisons :
 1. Intégration de données : les connecteurs permettent d'intégrer des données provenant de différentes sources pour les centraliser dans un système de stockage centralisé. Cela permet une analyse plus efficace des données.
 2. Automatisation des processus : les connecteurs peuvent automatiser les processus d'extraction, de transformation et de chargement de données, ce qui peut réduire les erreurs humaines et accélérer les délais de traitement des données.

3. Compatibilité avec différents systèmes : les connecteurs peuvent prendre en charge différents protocoles de communication et formats de données pour garantir une compatibilité avec les systèmes existants et les nouvelles sources de données.
 4. Flexibilité : les connecteurs peuvent être mis à jour ou remplacés pour prendre en charge de nouvelles sources de données ou des protocoles de communication plus récents, ce qui permet de maintenir la fiabilité et la flexibilité des processus ETL.
- **Transformateurs** : ce sont un composant clé des systèmes ETL, ils permettent de manipuler les données pour les adapter aux besoins de l'application de destination. Les transformateurs accomplissent des tâches telles que :
- Mapping : Les transformateurs effectuent le mapping des données en associant les champs ou attributs des sources de données aux champs correspondants de la destination. Cela garantit une correspondance correcte et cohérente des données lors de la transformation.
 - Règles de transformation : Les transformateurs appliquent des règles de transformation définies pour traiter les données extraites. Ces règles peuvent inclure des conversions de format, du filtrage, de l'agrégation, de la concaténation, de la normalisation et de la codification, entre autres. Les règles de transformation permettent d'adapter les données aux spécifications requises par l'application de destination.
1. Conversion de format : les transformateurs peuvent convertir les données de leur format d'origine en un format compatible avec la base de données cible.
 2. Filtrage : les transformateurs peuvent filtrer les données en utilisant des critères spécifiés pour ne conserver que les données nécessaires.
 3. Agrégation : les transformateurs peuvent regrouper les données en utilisant des critères spécifiés pour produire des statistiques ou des sommaires de données.
 4. Concaténation : les transformateurs peuvent concaténer les données provenant de plusieurs sources pour produire une source de données unique.
 5. Normalisation : les transformateurs peuvent normaliser les données pour éviter les redondances et garantir la qualité des données.
 6. Codification : les transformateurs peuvent convertir les données brutes en codes numériques ou textuels pour faciliter leur analyse.

En utilisant les transformateurs, les systèmes ETL peuvent garantir que les données sont préparées et prêtes à être utilisées efficacement par les applications de destination.

2.4.3 Exemples d'usage

Les systèmes ETL sont utilisés dans de nombreuses industries pour intégrer et exploiter les données pour diverses applications. Voici quelques exemples d'utilisation des ETL :

- Banques et finance : Les banques utilisent des systèmes ETL pour intégrer les données de différents systèmes pour une analyse financière plus complète et une meilleure prise de décision.
- Commerce de détail : Les entreprises de commerce de détail utilisent des systèmes ETL pour intégrer les données de différentes sources, telles que les systèmes de caisse, les systèmes de gestion de stock et les systèmes de gestion de la clientèle, pour une analyse de la performance plus complète.
- Santé : Les systèmes ETL sont utilisés dans le secteur de la santé pour intégrer les données de différents systèmes, tels que les dossiers médicaux électroniques, les systèmes de gestion des pharmacies et les systèmes de gestion des assurances, pour une analyse plus complète de la santé des patients.

- Marketing : Les entreprises de marketing utilisent des systèmes ETL pour intégrer les données de différentes sources, telles que les données démographiques, les données de vente et les données de marketing en ligne, pour une analyse plus complète des tendances du marché et de la performance de leurs campagnes de marketing.

2.5 Étude des outils

Dans le cadre de notre projet d'intégration de données dans le domaine de la santé, nous avons étudié plusieurs outils pour extraire, nettoyer, transformer et charger les données cliniques. Parmi ces outils, nous avons porté une attention particulière à trois outils principaux : Apache NiFi, OpenRefine et Trifacta.

2.5.1 Trifacta

i C'est quoi trifacta ?

Trifacta est une plateforme de préparation de données qui aide les utilisateurs à nettoyer, transformer et enrichir leurs données de manière efficace et rapide. Trifacta utilise des techniques de machine learning et d'interface utilisateur simplifiée pour automatiser et simplifier les tâches fastidieuses de préparation des données, permettant aux utilisateurs de se concentrer sur l'analyse et la prise de décision. La plateforme prend en charge une large gamme de sources de données et de formats de fichier, et fournit des fonctionnalités telles que la détection automatique de schéma, la suggestion de transformation et l'aperçu en temps réel pour faciliter le processus de préparation des données. Trifacta est utilisé dans divers secteurs tels que la finance, la santé, les médias et les télécommunications pour améliorer la qualité et la précision des données et accélérer l'analyse de données.

ii Comment on peut s'en servir de Trifacta dans le domaine médical ?

Trifacta peut être utilisé pour préparer et nettoyer des données médicales dans divers contextes, tels que la recherche, les essais cliniques, la gestion des dossiers médicaux et l'analyse des données de santé.

La préparation des données médicales peut être un processus complexe et fastidieux, car les données peuvent être très volumineuses, provenir de différentes sources et être dans différents formats. De plus, les données médicales peuvent contenir des informations sensibles sur les patients, ce qui rend la gestion et le traitement des données encore plus complexes.

Trifacta peut aider les professionnels de la santé et les chercheurs à préparer leurs données plus rapidement et plus efficacement en utilisant des techniques d'automatisation et d'apprentissage automatique. Les fonctionnalités de Trifacta, telles que la détection automatique de schéma, la suggestion de transformation et l'aperçu en temps réel, peuvent aider à identifier et à nettoyer les données manquantes, dupliquées ou erronées, ainsi qu'à normaliser et à standardiser les données pour améliorer la qualité et la précision de l'analyse.

De plus, Trifacta dispose de fonctionnalités de sécurité et de confidentialité pour gérer les données sensibles, telles que la pseudonymisation des données, la gestion des accès et des autorisations et la suppression sécurisée des données.

Il convient toutefois de noter que le traitement des données médicales peut être soumis à des réglementations strictes en matière de confidentialité et de protection des données. Il est important

de respecter les réglementations en vigueur et de mettre en place des mesures de sécurité et de confidentialité appropriées pour protéger les données des patients.

iii Exemple d'utilisation

Pour nettoyer nos données sur Trifacta ,nous avons commencé par télécharger un fichier csv de données hospitalières relatives à l'épidémie de COVID-19 en France en 2020.

Ce dernier est constitué de 88145 lignes, contient comme colonnes : code du Département, date, nb actuellement hospitalisés, Nb actuellement en soins intensifs, total retour à domicile, total décès, Code région, Code Iso, Nom région, Nom département, Sexe, Geo_point_2d, HospConv,SSR_USLD, autres, Nb Quotidien Admis Réanimation, Nb Quotidien Décès, Nb Quotidien Retour à Domicile.

Aussi, Trifacta nous permet de visualiser les données à travers des statistiques. Ces statistiques incluent des informations telles que le nombre de lignes et de colonnes, le nombre de valeurs valides, le nombre de valeurs manquantes, le nombre de doublons et d'autres informations pertinentes. Ces statistiques nous aident à comprendre la qualité des données après leur nettoyage et à déterminer si les données sont prêtes pour l'analyse, voici l'exemple pour notre cas.



FIGURE 1 – Résultats petit échantillon

Comme nous pouvons voir, nous avons 71% de valeurs valides, 0.1% de valeurs manquantes et 28% de valeurs erronées, cela signifie que la plupart des données sont valides et utilisables, mais qu'il y a un certain nombre de valeurs erronées qui nécessitent d'être corrigées pour maximiser la précision et la fiabilité de l'analyse qui sera effectuée sur ces données.

Ensuite, pour comprendre le contenu des données et avoir une idée de leur qualité globale, une méthode a consisté à observer les données. Pour identifier les zones nécessitant une correction, les anomalies, les incohérences, les doublons et les valeurs erronées ont été identifiés. Cette étape d'observation a été essentielle pour garantir que les données soient nettoyées avec précision et que toutes les erreurs potentielles aient été corrigées.

Les étapes de traitement utilisées sont :

- Remplacement des valeurs manquantes par des zéros dans quatre colonnes qui étaient de type 'int'.
- remplacement des valeurs '2B' et '2A' qui représentent respectivement les codes de département Corse Sud et Corse Nord dans la colonne 'code département' (de type 'int') par la valeur '20'. Ces deux codes sont spécifiques à la Corse et ne sont pas reconnus par certains logiciels d'analyse, d'où la nécessité de les remplacer par une valeur standard.
- Remplacement des valeurs manquantes dans la colonne "nom région" par la valeur "null"
- Suppression des lignes en double pour éviter toute duplication de données.
- Suppression de trois lignes qui étaient vides et qui n'avaient aucune information utile, notamment sont HosConv, SSR_USLD, autres.
- Suppression des lignes contenant des valeurs manquantes dans la colonne "geo_point_2d"
- Remplacement des valeurs manquantes dans la colonne "nom département" par la valeur "null"

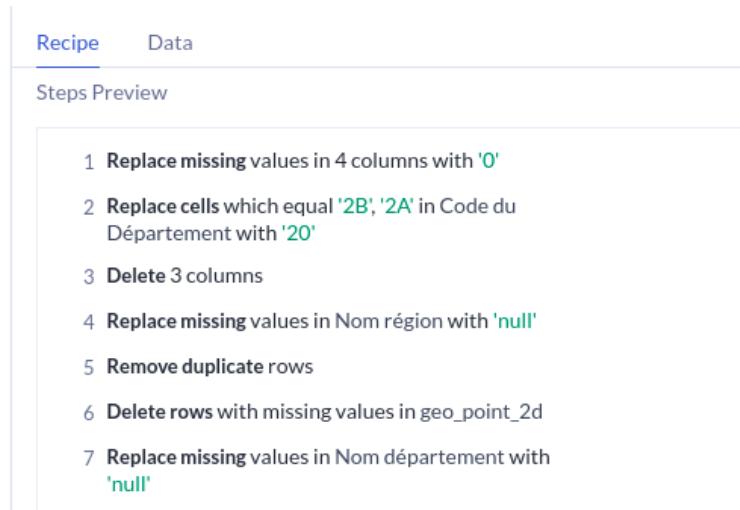


FIGURE 2 – Image montrant les différentes étapes

Au final, Triflows nous permet de visualiser les différentes étapes de nettoyage des données que nous avons effectuées sous forme d'un schéma final. Ce schéma nous permet de mieux comprendre comment les différentes actions ont été appliquées aux données et comment elles ont affecté les différentes colonnes. Cela nous permet également de voir si toutes les actions ont été effectuées avec succès et de vérifier si les données ont été correctement nettoyées.

Voici le schéma obtenu pour notre exemple :

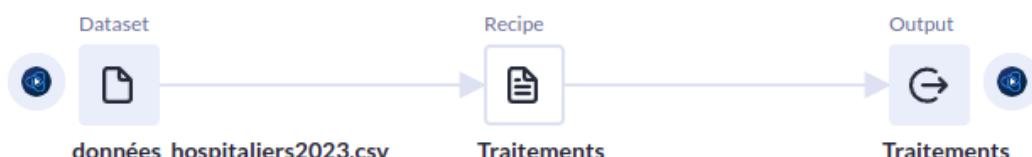


FIGURE 3 – Schème final

iv Résultats finaux

Après avoir effectué les différentes étapes de nettoyage de données à l'aide de Triflows, on peut remarquer que les résultats finaux sont satisfaisants. Les données sont maintenant valides à 100%, il n'y a plus de valeurs manquantes ni de valeurs erronées. Cela signifie que les données sont maintenant prêtes à être utilisées pour des analyses ultérieures ou des visualisations.

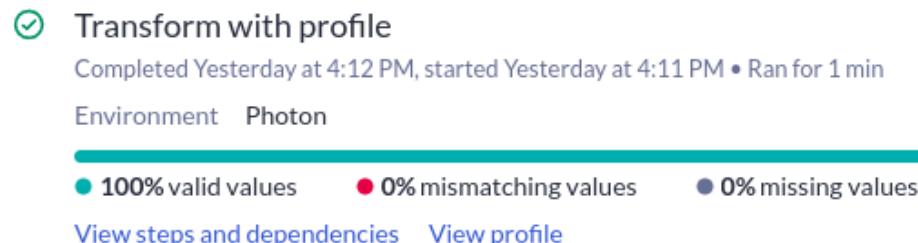


FIGURE 4 – Résultats

En outre, ces résultats finaux nous montrent l'efficacité des étapes de nettoyage de données que nous avons effectuées à l'aide de Triflacta pour améliorer la qualité de nos données.

Results profile by column

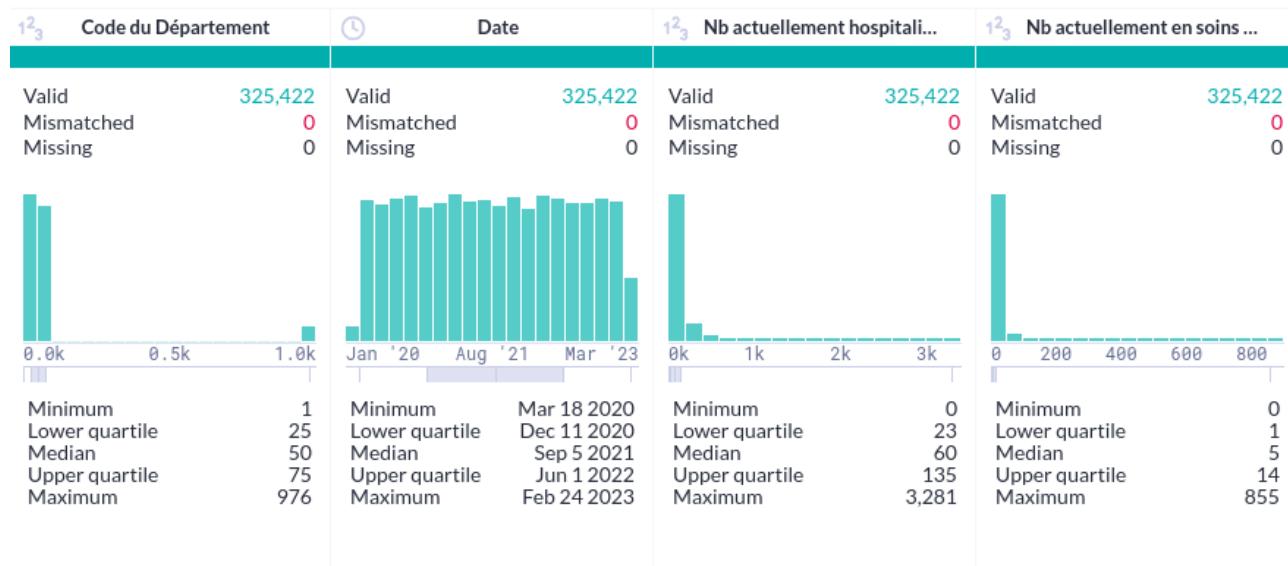


FIGURE 5 – statistiques de quelques colonnes

Voici un exemple de statistiques pour quelques colonnes qui montre que nous avons 325 422 données valides, aucune valeur ne correspond pas et aucune valeur manquante.

En résumé, Triflacta offre des avantages significatifs dans le domaine de la préparation des données, notamment en simplifiant et en accélérant le processus, en améliorant la qualité des données et en facilitant la compréhension des informations. Cependant, l'utilisation de Triflacta peut entraîner des coûts d'abonnement et nécessite une courbe d'apprentissage initiale. De plus, il est important de mettre en place des mesures de sécurité et de confidentialité adéquates pour protéger les données sensibles conformément aux réglementations en vigueur.

2.5.2 OpenRefine

OpenRefine est un outil open-source et gratuit conçu pour faciliter le nettoyage, la transformation et l'exploration de données. Il est particulièrement utile pour traiter des ensembles de données com-

plexes et désordonnés, en offrant des fonctionnalités avancées pour la normalisation, la correction d'erreurs, la fusion et la manipulation de données.

OpenRefine offre une interface conviviale qui permet aux utilisateurs de charger des ensembles de données brutes, qu'ils soient au format CSV, TSV, Excel ou JSON. L'outil analyse automatiquement les données pour détecter les incohérences, les doublons et les erreurs courantes, facilitant ainsi le processus de nettoyage initial.

Une fois les données chargées, OpenRefine propose une gamme de transformations puissantes pour structurer et normaliser les données. Les utilisateurs peuvent appliquer des opérations de filtrage, de tri, de fractionnement, de fusion et de regroupement pour organiser et segmenter les données en fonction de leurs besoins spécifiques.

Une autre fonctionnalité puissante d'OpenRefine est sa capacité à effectuer des opérations de nettoyage avancées à l'aide de fonctions de recherche et de remplacement, d'expressions régulières et de règles de clustering. Ces outils permettent de résoudre des problèmes courants tels que la correction des fautes de frappe, la normalisation des formats de date, l'unification des valeurs similaires et la suppression des valeurs aberrantes.

OpenRefine offre également des fonctionnalités d'exploration de données pour faciliter la visualisation et l'analyse des ensembles de données. Les utilisateurs peuvent générer des statistiques, des facettes, des graphiques et des résumés interactifs pour mieux comprendre la distribution et les caractéristiques des données.

L'un des avantages majeurs d'OpenRefine est sa capacité à enregistrer toutes les étapes de nettoyage et de transformation effectuées sur les données, créant ainsi un flux de travail reproductible. Cela permet aux utilisateurs de documenter et de partager facilement leurs processus de nettoyage avec d'autres membres de l'équipe ou la communauté.

Il s'agit de soumettre un fichier en entrée dans l'un des formats autorisés, effectuer des traitements avec les opérations possibles sur le logiciel ou par l'exécution de scripts et obtenir un fichier en sortie.

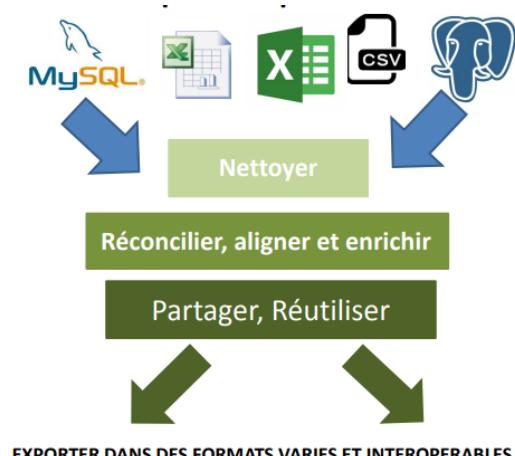


FIGURE 6 – Schéma récapitulatif du fonctionnement de open refine

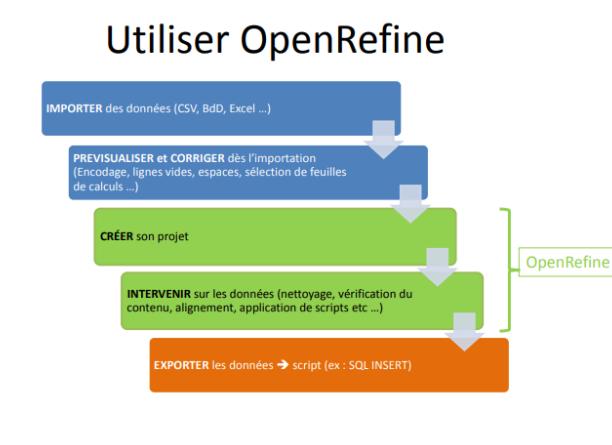


FIGURE 7 – Cycle de la transformation des données

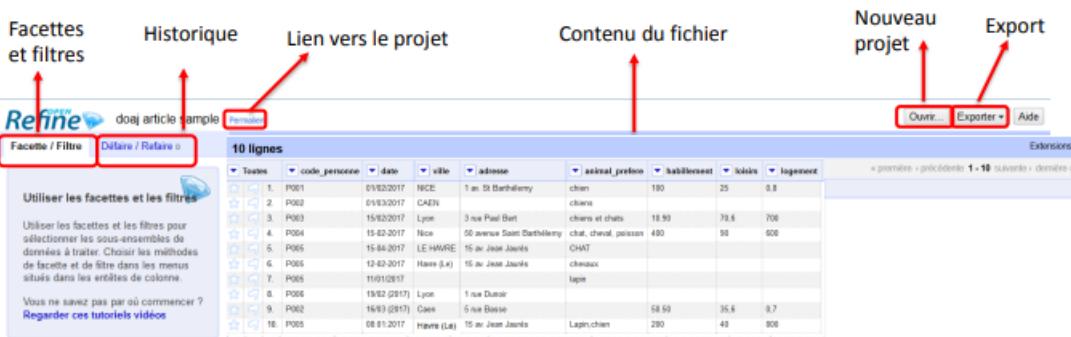


FIGURE 8 – Présentation de l'espace de travail

Bien qu'OpenRefine soit un outil largement utilisé pour le nettoyage et la transformation des données, il présente certains désavantages à prendre en considération. Tout d'abord, il utilise une interface graphique, ce qui limite certaines possibilités d'automatisation et d'application d'opérations simultanément. De plus, son modèle de données est contraint à un format tabulaire, ce qui peut restreindre sa flexibilité pour des types de données plus complexes. OpenRefine n'est pas adapté au traitement de "big data" en raison de problèmes de performance rencontrés lors du traitement de grandes quantités de données. De plus, il ne permet pas le traitement de données en flux ou en temps réel. L'utilisation de OpenRefine peut également nécessiter une quantité importante de mémoire, ce qui peut être problématique lorsque les ressources sont limitées. En outre, OpenRefine ne dispose pas de fonctionnalités collaboratives avancées, limitant ainsi la possibilité de collaboration entre utilisateurs sur un même projet. Cependant, malgré ces désavantages, OpenRefine reste un outil puissant et largement utilisé pour le nettoyage et la transformation des données, adapté à de nombreux cas d'utilisation.

2.5.3 Étude de NiFi

i Présentation de l'outil

Le troisième outil testé est NiFi. Apache NiFi est une plateforme open-source puissante et polyvalente conçue pour faciliter l'automatisation et la gestion des flux de données dans des environnements complexes. Grâce à son architecture flexible et évolutive, NiFi offre des fonctionnalités avancées pour l'extraction, la transformation et le chargement (ETL) des données, ainsi que pour l'intégration de systèmes hétérogènes.

L'un des atouts majeurs de NiFi réside dans sa capacité à fournir une interface graphique intuitive basée sur un modèle de programmation visuel. Cela permet aux utilisateurs de concevoir et de configurer des flux de données en les reliant simplement à l'aide de composants prédéfinis appelés processeurs. Les processeurs représentent des opérations spécifiques, telles que la lecture de fichiers, l'extraction de données, la transformation, le filtrage, l'enrichissement et le chargement dans des systèmes cibles. Les flux de données créés peuvent être facilement adaptés, ajustés et surveillés, offrant ainsi une flexibilité et une visibilité accrues sur les opérations de traitement des données.

NiFi prend en charge de nombreux protocoles et formats de données couramment utilisés, permettant ainsi l'intégration transparente de sources et de destinations variées. Il peut se connecter à des bases de données relationnelles, des services web, des files d'attente de messages, des systèmes de

fichiers, des services cloud, des applications IoT, et bien plus encore. NiFi offre également des fonctionnalités avancées pour la gestion des erreurs et la reprise sur panne, garantissant ainsi l'intégrité et la fiabilité des flux de données même dans des environnements instables ou soumis à des interruptions.

La sécurité des données est une priorité dans NiFi. Il propose des fonctionnalités de chiffrement, d'authentification, d'autorisation et de journalisation pour protéger les données sensibles et garantir la conformité aux réglementations de confidentialité. Les contrôles d'accès granulaires permettent de définir des politiques de sécurité personnalisées pour restreindre l'accès aux données et aux opérations critiques.

ii Utilisation de Nifi dans un contexte hospitalier

Dans le domaine hospitalier, Apache NiFi peut être utilisé pour automatiser et gérer le flux de données cliniques entre différentes sources et destinations, améliorant ainsi l'efficacité opérationnelle et la qualité des soins. Voici un exemple concret d'utilisation de NiFi dans un environnement hospitalier : Imaginons un hôpital qui utilise différents systèmes pour collecter des données médicales, tels que les dossiers médicaux électroniques, les dispositifs de surveillance, les résultats de laboratoire et les images médicales. Chaque système génère des données dans un format spécifique et les stocke localement. Grâce à Apache NiFi, l'hôpital peut mettre en place un flux de données automatisé pour intégrer ces différentes sources de données. Les processeurs NiFi peuvent être configurés pour extraire les données des différents systèmes, les transformer en un format standardisé et les charger dans un entrepôt de données centralisé. Par exemple, le flux de données peut être conçu pour extraire les résultats de laboratoire des fichiers CSV produits par le système de laboratoire, extraire les données vitales des dispositifs de surveillance via des protocoles de communication standard tels que HL7, et extraire les rapports d'imagerie médicale à partir du système d'imagerie DICOM.

Une fois les données extraites, les processeurs de transformation de NiFi peuvent être utilisés pour nettoyer, enrichir et normaliser les données. Par exemple, les valeurs aberrantes peuvent être filtrées, les identifiants de patients peuvent être anonymisés pour assurer la confidentialité, et les données peuvent être enrichies avec des informations supplémentaires provenant d'autres sources.

Enfin, les données transformées peuvent être chargées dans un entrepôt de données centralisé, tel que PostgreSQL ou MongoDB, où elles peuvent être exploitées pour des analyses ultérieures, des rapports cliniques ou des applications de prise de décision.

iii Les différents processeurs nifi

Un processeur NiFi est un composant essentiel de la plateforme Apache NiFi, un système de gestion de données en temps réel et de flux de données. Un processeur NiFi est une unité de traitement qui permet d'effectuer diverses opérations sur les flux de données, telles que la lecture, la transformation, la validation, l'enrichissement et la distribution des données. Chaque processeur exécute une tâche spécifique et peut être connecté à d'autres processeurs pour créer des flux de données complexes et personnalisés. Voici quelques-uns des processeurs NiFi les plus importants :

- **GetFile** : Ce processeur permet de lire des fichiers à partir d'un répertoire local ou d'un système de fichiers distant. Il récupère les données et les envoie au flux de données NiFi pour un traitement ultérieur, les propriétés les plus importantes de ce dernier dans Apache NiFi sont les suivantes :
 - Directory : Le répertoire à partir duquel les fichiers doivent être lus.
 - Recursive : Indique si la recherche des fichiers doit être récursive dans les sous-répertoires.
 - File Filter : Permet de spécifier un filtre pour sélectionner uniquement certains fichiers.

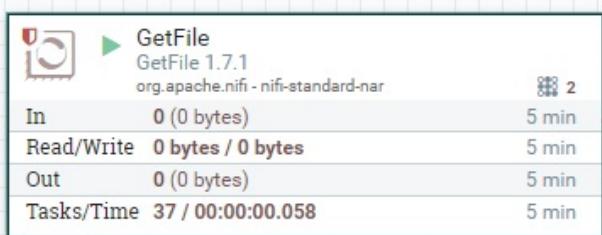


FIGURE 9 – processeur "Getfile"

Processor Details					
		SETTINGS	SCHEDULING	PROPERTIES	COMMENTS
Required field					
Property					Value
Input Directory	●	C:\input			
File Filter	●	!*.J.*			
Path Filter	●	No value set			
Batch Size	●	10			
Keep Source File	●	false			
Recurse Subdirectories	●	true			
Polling Interval	●	0 sec			
Ignore Hidden Files	●	true			
Minimum File Age	●	0 sec			
Maximum File Age	●	No value set			
Minimum File Size	●	0 B			
Maximum File Size	●	No value set			

FIGURE 10 – Propriétés du Getfile

- **PutFile** : Ce processeur permet d'écrire des données dans des fichiers sur un système de fichiers local ou distant. Il est utilisé pour stocker les résultats ou les données transformées dans des fichiers persistants, les trois propriétés les plus importantes de ce dernier dans Apache NiFi sont les suivantes :
 - Directory : Le répertoire dans lequel les fichiers doivent être écrits.
 - Conflict Resolution : Détermine comment gérer les conflits de noms de fichiers existants.
 - Append : Indique si les données doivent être ajoutées à des fichiers existants ou s'ils doivent être remplacés.

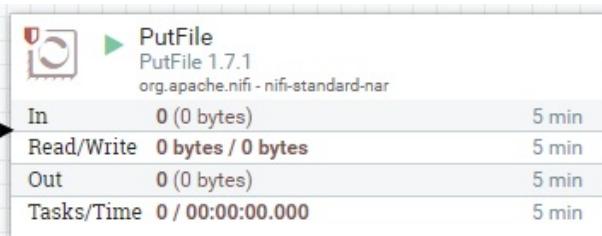


FIGURE 11 – Processeur "Putfile"

Processor Details					
		SETTINGS	SCHEDULING	PROPERTIES	COMMENTS
Required field					
Property					Value
Directory	●	C:\output			
Conflict Resolution Strategy	●	fail			
Create Missing Directories	●	true			
Maximum File Count	●	No value set			
Last Modified Time	●	No value set			
Permissions	●	No value set			
Owner	●	No value set			
Group	●	No value set			

FIGURE 12 – Propriétés du Putfile

- **ExecuteStreamCommand** : Ce processeur permet d'exécuter des commandes système ou des scripts externes à l'aide de la ligne de commande dans un flux de données. Les propriétés les plus importantes de ce processeur dans Apache NiFi sont les suivantes :
 - Working Directory (Répertoire de travail) : spécifie le répertoire de travail dans lequel la commande sera exécutée. Cela peut être utile si la commande ou le script dépend d'un certain contexte de répertoire.
 - Command Path (Chemin de la commande) : indique le chemin complet vers l'exécutable de la commande ou du script que vous souhaitez exécuter. Vous devez spécifier le chemin complet de la commande, y compris le nom de l'exécutable.
 - Command Arguments (Arguments de la commande) : permet de spécifier les arguments supplémentaires à passer à la commande ou au script lors de l'exécution.
 - Command Timeout (Délai d'attente de la commande) : définit le délai d'attente maximal autorisé pour l'exécution de la commande, après quoi elle sera considérée comme ayant

échoué.

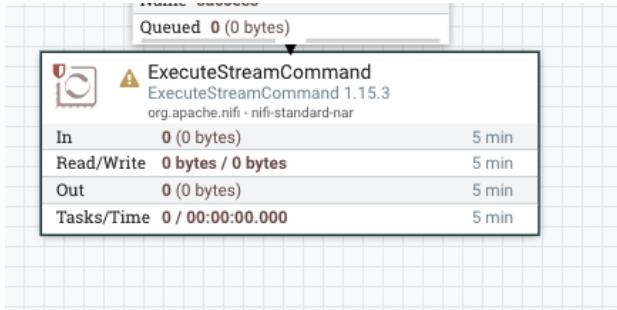


FIGURE 13 – processeur "ExecuteStreamCommand"

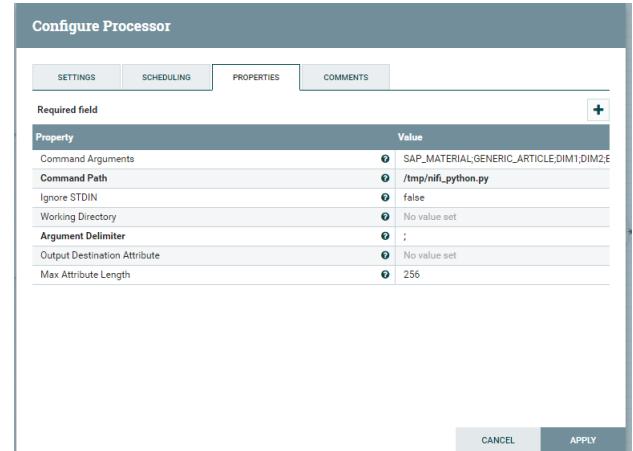


FIGURE 14 – Propriétés de ExecuteStreamCommand

- **ReplaceText** : Ce processeur permet de remplacer des valeurs de texte dans les flux de données. Il est couramment utilisé pour nettoyer les données en supprimant les caractères spéciaux, les espaces indésirables ou en appliquant des expressions régulières., les propriétés les plus importantes de ce dernier dans Apache NiFi sont les suivantes :
 - Search Value (Valeur de recherche) : spécifie la chaîne de caractères que nous souhaitons rechercher dans le contenu de vos flux de données.
 - Replacement Value (Valeur de remplacement) : spécifie la chaîne de caractères par laquelle nous souhaitons remplacer la valeur de recherche trouvée.
 - Replacement Strategy (Stratégie de remplacement) : définit la stratégie de remplacement à utiliser, telle que "Remplacement littéral" pour un remplacement direct ou "Remplacement basé sur une expression régulière" pour un remplacement en utilisant des expressions régulières.

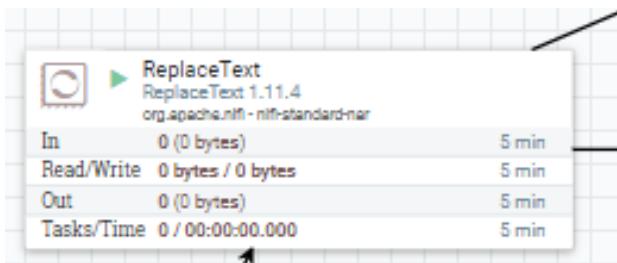


FIGURE 15 – processeur "Replace text"

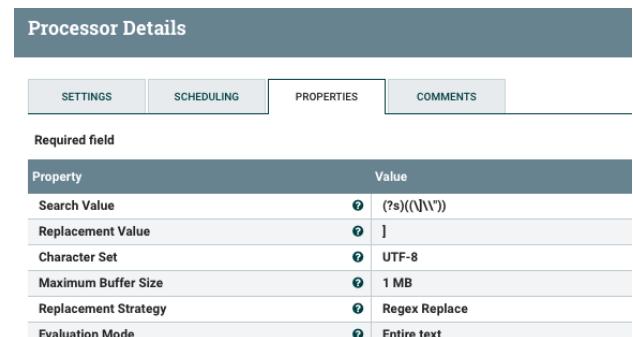


FIGURE 16 – Propriétés de Replace text

- **QueryRecord** : Ce processeur permet d'exécuter des requêtes SQL sur les flux de données. Il est utilisé pour effectuer des opérations de filtrage, de jointure ou de regroupement des données, les propriétés les plus importantes de ce dernier dans Apache NiFi sont les suivantes :
 - Record Reader (Lecteur d'enregistrements) : spécifie le lecteur d'enregistrements à utiliser pour lire les données d'entrée.
 - Record Writer (Écrivain d'enregistrements) : spécifie l'écrivain d'enregistrements à utiliser pour écrire les données de sortie. SQL Query (Requête SQL) : définit la requête SQL à exécuter sur les enregistrements d'entrée.

- Schema Registry (Registre de schémas) : spécifie le registre de schémas à utiliser pour gérer les schémas des enregistrements.

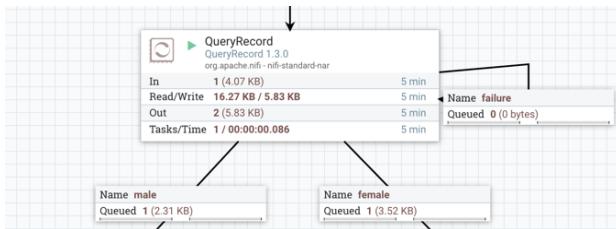


FIGURE 17 – processeur "Query record"

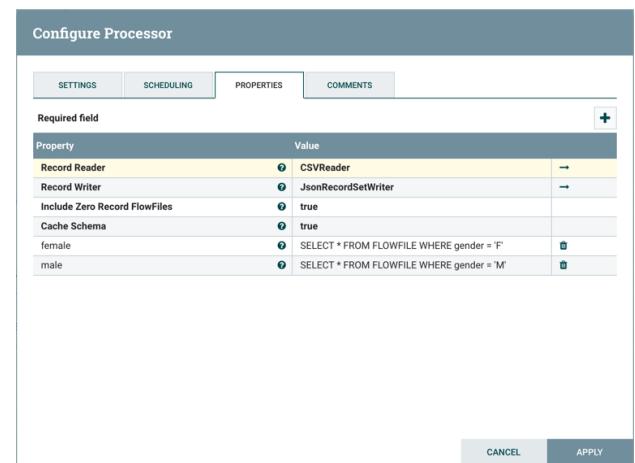


FIGURE 18 – Propriétés de Query record

- **SplitText** : Ce processeur divise un flux de données en fonction d'un délimiteur spécifié, ce qui permet de traiter chaque partie individuellement. Il est utile lorsque les données sont structurées en blocs ou enregistrements distincts, les propriétés les plus importantes de ce dernier dans Apache NiFi sont les suivantes :
 - Line Split Count (Nombre de lignes à diviser) : cette propriété spécifie le nombre de lignes à extraire à partir du contenu textuel d'un flux de données.
 - Header Line Count (Nombre de lignes d'en-tête) : indique le nombre de lignes d'en-tête à ignorer avant de commencer à diviser le texte.
 - Footer Line Count (Nombre de lignes de pied de page) : Cette propriété spécifie le nombre de lignes de pied de page à ignorer après avoir divisé le texte.
 - Split Strategy (Stratégie de division) : Elle détermine la stratégie de division à appliquer, par exemple, "Diviser toutes les lignes" pour diviser chaque ligne individuellement ou "Diviser en lots" pour diviser le texte en lots de lignes spécifiées.

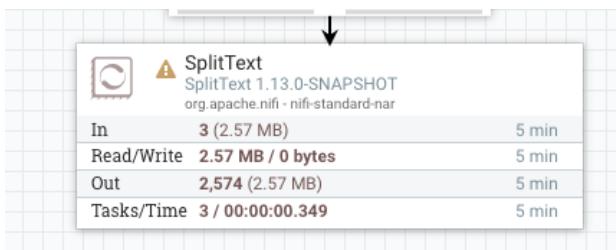


FIGURE 19 – processeur "Split text"

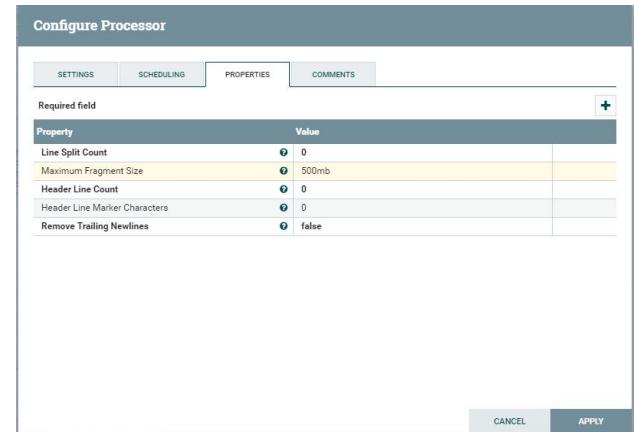


FIGURE 20 – Propriétés de Split text

- **MergeContent** : Ce processeur fusionne plusieurs flux de données en un seul flux. Il est utilisé pour regrouper les données provenant de différentes sources ou pour combiner des flux de données parallèles, les propriétés les plus importantes de ce dernier dans Apache NiFi sont les suivantes :
 - Merge Format (Format de fusion) : spécifie le format dans lequel les contenus fusionnés doivent être combinés.

- Minimum Number of Entries (Nombre minimum d'entrées) : définit le nombre minimum d'entrées requis pour déclencher la fusion.
- Maximum Number of Entries (Nombre maximum d'entrées) : spécifie le nombre maximum d'entrées à fusionner dans un seul flux de sortie.
- Maximum Number of Bins (Nombre maximum de bacs) : contrôle le nombre maximum de bacs utilisés pour stocker temporairement les entrées lors du processus de fusion.
- Merge Strategy (Stratégie de fusion) : permet de choisir la stratégie de fusion à appliquer lors de la combinaison des contenus.

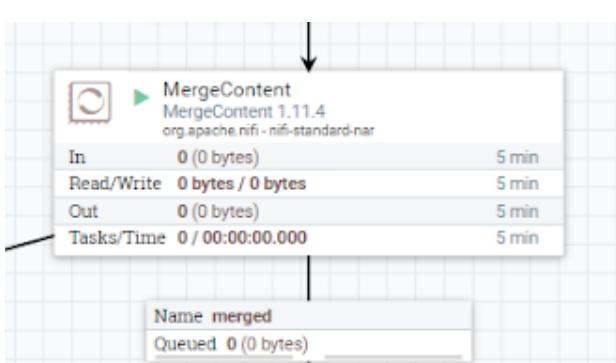


FIGURE 21 – processeur "MergeContent"

Property	Value
Merge Strategy	Bin-Packing Algorithm
Merge Format	Binary Concatenation
Attribute Strategy	Keep Only Common Attributes
Correlation Attribute Name	UUID
Minimum Number of Entries	2
Maximum Number of Entries	2
Minimum Group Size	0 B
Maximum Group Size	No value set
Max Bin Age	No value set
Maximum number of Bins	100
Delimiter Strategy	Text
Header	[
Footer]
Name suffix	

FIGURE 22 – Propriétés de Merge-Content

Ces processeurs NiFi offrent une flexibilité et une extensibilité considérables pour le traitement des flux de données en temps réel. Ils peuvent être combinés et configurés de différentes manières pour répondre aux besoins spécifiques des pipelines de données et des cas d'utilisation variés.

iv Fichier de configuration Nifi Properties

```
#####
nifi.web.https.host=127.0.0.1
nifi.web.https.port=8443
nifi.web.https.network.interface.default=
nifi.web.https.application.protocols=http/1.1
nifi.web.jetty.working.directory=./work/jetty
nifi.web.jetty.threads=200
nifi.web.max.header.size=16 KB
nifi.web.proxy.context.path=
nifi.web.proxy.host=
nifi.web.max.content.size=
nifi.web.max.requests.per.second=30000
nifi.web.max.access.token.requests.per.second=25
nifi.web.request.timeout=60 secs
nifi.web.request.ip.whitelist=
nifi.web.should.send.server.version=true
nifi.web.request.log.format=%{client}a - %u %t "%r" %s %O "%{Referer}i" "%{User-Agent}i"
```

FIGURE 23 – Aperçu du fichier config Nifi

Le fichier de configuration de NiFi, appelé nifi.properties, est un fichier utilisé pour spécifier comment NiFi doit se comporter et quelles options il doit utiliser. Il contient différentes sections et paramètres qui contrôlent différents aspects du fonctionnement de NiFi. Voici quelques-unes des sections et des paramètres clés que l'on peut trouver dans le fichier de configuration :

- Section de configuration globale : Cette section permet de spécifier des paramètres généraux pour NiFi, tels que le chemin du fichier de configuration du flux de données NiFi et si les versions précédentes du flux de données doivent être archivées.
- Section de sécurité : Cette section est utilisée pour configurer les paramètres de sécurité de NiFi, tels que les chemins des fichiers de clés et de certificats utilisés pour sécuriser les communications.
- Section du serveur web embarqué : Cette section concerne le serveur web intégré à NiFi. Elle permet de spécifier l'adresse IP ou le nom d'hôte sur lequel le serveur web écoute les requêtes HTTP, ainsi que le port sur lequel il écoute.
- Section des journaux : Cette section concerne les journaux de NiFi. Elle permet de spécifier le répertoire dans lequel les journaux doivent être enregistrés, ainsi que le niveau de détail des journaux.
- Section de base de données : Cette section est utilisée pour configurer les paramètres relatifs à la base de données interne de NiFi. Elle spécifie le répertoire dans lequel les bases de données doivent être stockées.

2.5.4 Comparaison et choix des outils

Maintenant que tous les concepts nécessaires à la compréhension des tâches globales ont été introduits ,nous pouvons entamer la comparaison des trois outils testés. Nous avons d'abord travaillé avec Trifacta, qui nous propose une interface simple et conviviale. Les traitements sur les colonnes se fait de manière très intuitive. Ce fut assez simple d'utilisation. Seulement, certaines transformations sont impossibles à l'aide de cet outil. On a dû se rediriger vers un autre outil, OpenRefine qui permettait quant à lui d'effectuer plus de transformations. En effet, on a pu appliquer toutes nos règles de transformation grâce à OpenRefine, soit en utilisant des options disponibles sur le logiciel, soit par l'usage de scripts écrits en GREL. Les expressions GREL sont utilisées pour modifier les cellules dans une colonne, créer de nouvelles colonnes en utilisant des formules, filtrer les lignes de données en fonction de critères, et bien plus encore. Les fonctions GREL incluent des fonctions de manipulation de chaînes de caractères, de dates, de nombres, de listes et de tableaux. Il existe également des fonctions pour faire des calculs mathématiques, des comparaisons logiques, des recherches de texte et des extractions de sous-chaînes. Par contre, nous n'avons pas pu effectuer le mapping sur OpenRefine . Ce que nous avons fait, par conséquent, c'est que nous avons effectué le mapping sur Nifi et nous avons codé les règles sur OpenRefine.

Nous avons cherché une manière de lier OpenRefine et Nifi, pour pouvoir effectuer des travaux distincts et complémentaires. Seulement, nous avons compris que cela n'était pas possible.

nous avons donc dans un premier temps, effectué tous les prétraitements liés au pré-nettoyage des données sur OpenRefine, avant de passer le fichier résultant à Nifi pour réaliser le mapping puis charger le fichier sortant dans openRefine pour appliquer tous les prétraitements relatifs aux règles. Avant de, dans un second temps, décider de tout effectuer sur le même logiciel. Nous avons remarqué que l'outil qui permettait de réaliser tout le travail demandé en même temps était Nifi. Par ailleurs, nous notons que la transformation via Nifi est plus optimale que la transformation via OpenRefine dans la mesure où, sur OpenRefine, à chaque transformation à partir d'un fichier, nous devons ouvrir un nouveau projet openrefine et exporter les règles que nous avons défini précédemment sur un autre fichier. Tandis que sur Nifi, nous restons sur le même projet en modifiant uniquement le choix du fichier qui doit être transformé.

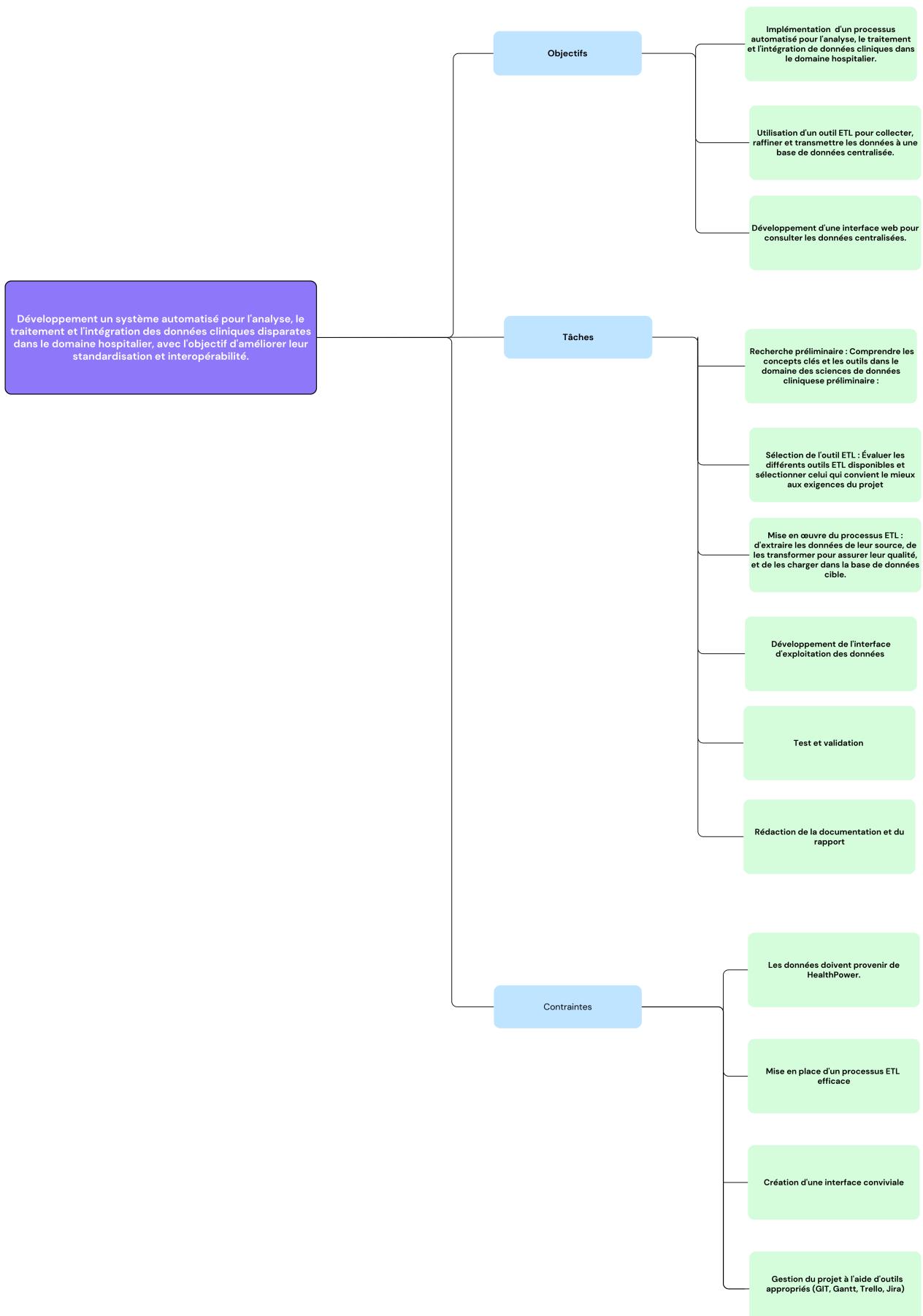
A la lumière de nos expérimentations, nous attestons que l'outil le plus approprié pour effectuer le travail demandé est Nifi.

3 Approche méthodologique

3.1 Cahier de charges

Avant de commencer la réalisation du projet, il est essentiel de comprendre le cadre de référence de ce projet. Le schéma qui suit présente le cahier des charges détaillé que nous avons élaboré pour guider la réalisation de notre projet. Ce document est une pièce maîtresse de la conception, car il définit précisément les objectifs à atteindre, les spécifications techniques ainsi que les contraintes à respecter.

Le cahier des charges détaillé est présenté ci-dessous :



3.2 Conception du projet

La conception de ce projet d'intégration des données cliniques a été guidée par plusieurs objectifs principaux. Tout d'abord, nous avons cherché à mettre en place un processus d'intégration robuste et efficace, capable de gérer de grands volumes de données cliniques hétérogènes. Après quoi, nous avons cherché à garantir que les données intégrées soient précises, à jour et conformes aux normes de confidentialité et de sécurité.

3.2.1 Architecture du système

L'architecture de notre système est basée sur une approche d'intégration de données en trois étapes : extraction, transformation et chargement (ETL). Les données sont d'abord extraites de diverses sources, y compris les systèmes de dossiers de santé électroniques (EHR), les bases de données de laboratoire, et autres sources de données cliniques. Ensuite, ces données sont transformées pour garantir leur cohérence et leur compatibilité, avant d'être chargées dans une base de données centrale pour une analyse ultérieure.

3.2.2 Sélection des outils

Nous avons sélectionné un ensemble d'outils (tels que NiFi, OpenRefine...) pour soutenir notre processus d'intégration de données. Ces outils ont été choisis en fonction de leur capacité à gérer de grands volumes de données, leur conformité avec les normes de sécurité des données de santé, et leur facilité d'utilisation et de maintenance.

3.2.3 Processus ETL

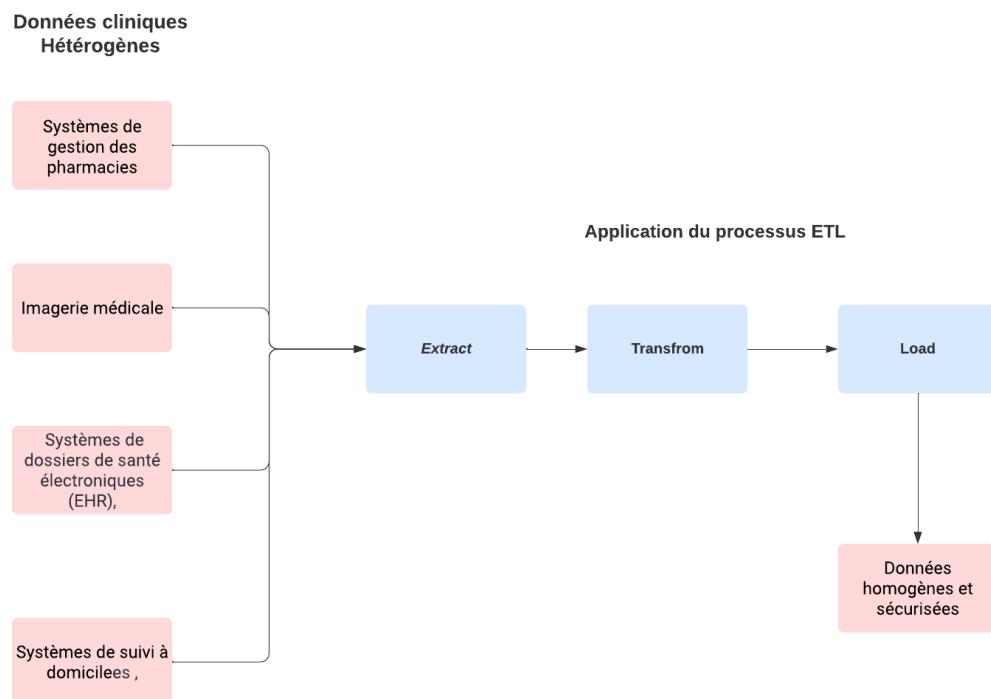
Un processus ETL détaillé a été conçu pour guider l'intégration des données. Ce processus comprend des procédures pour l'extraction des données à partir de sources hétérogènes, la transformation des données pour assurer leur compatibilité, et le chargement des données dans la base de données centrale.

3.2.4 Interface utilisateur

Enfin, une interface utilisateur a été conçue pour permettre aux utilisateurs d'interagir facilement avec le système. Cette interface comprend des fonctionnalités pour la visualisation des données, la gestion des tâches d'intégration de données, et l'interaction avec les résultats de l'analyse.

3.2.5 Diagramme de l'architecture du système

Pour mieux illustrer le processus d'intégration de données décrit ci-dessus, nous avons conçu un schéma qui démontre les différentes étapes du processus d'ETL. Ce schéma permet de visualiser comment les données sont extraites de sources hétérogènes, transformées pour assurer leur compatibilité, puis chargées.



Le schéma illustre le processus d'ETL (Extraction, Transformation, Chargement) utilisé pour intégrer les données cliniques dans notre système. Il commence par l'extraction des données de diverses sources hétérogènes, comme les dossiers de santé électroniques, les bases de données de laboratoire et d'autres sources cliniques pertinentes. Ces données sont ensuite transformées pour assurer leur cohérence et leur compatibilité. L'interface utilisateur, qui est un élément clé de notre système, joue un rôle essentiel à cette étape, permettant aux utilisateurs de gérer et de contrôler le processus d'intégration des données. Enfin, les données transformées sont chargées dans la base de données centrale pour une analyse ultérieure. Ce processus garantit que nos données intégrées soient précises, à jour et conformes aux normes de confidentialité et de sécurité.

Nous détaillons plus amplement chaque étape de la conception au niveau de l'axe suivant.

4 Réalisation et résultats

4.1 Mise en œuvre du processus ETL pour le traitement de données cliniques

Dans cette section, nous abordons le processus ETL (Extraction, Transformation et Chargement) et sa mise en œuvre pour intégrer des données dans le domaine de la santé. Nous détaillons les différentes étapes et présentons des exemples pour illustrer notre propos.

4.1.1 Avant de commencer : Anonymisation des données

i Contexte

L'anonymisation des données est un processus qui a pour objectif de protéger la confidentialité des sujets représentés dans une base de données. Dans le cas des données médicales, cette procédure est

cruciale afin de respecter la vie privée des patients ainsi que pour se conformer aux réglementations en vigueur concernant la protection des données à caractère personnel.

Dans ce projet, nous avons effectué l'anonymisation d'un ensemble de données de santé provenant de l'organisation PowerHealth. Le but était de remplacer les informations identifiables par des informations génériques, tout en conservant la structure et l'utilité des données pour les analyses futures.

ii Méthodologie

Nous avons développé un script en Python qui utilise la bibliothèque pandas pour lire et manipuler les données, ainsi que les bibliothèques random et names pour générer des données anonymes. Ce même script va être appelé sur Nifi grâce au processeur ExecuteStreamCommand qui va faciliter la tâche et la rendre très rapide (vu qu'on travaille avec des données volumineuses ≈ 100000000 lignes), Le script fonctionne en suivant ces étapes :

- Lecture du fichier Excel contenant les données à anonymiser. Le fichier est lu feuille par feuille en créant un DataFrame pour chaque feuille.
- Anonymisation des données :
 - Les noms des patients, les noms des consultants et les noms des hôpitaux sont remplacés par des noms générés aléatoirement.
 - Les numéros d'identification des patients, les identifiants de rencontre, les numéros MR et les identifiants des médecins sont remplacés par des numéros générés aléatoirement.
 - Nous avons pris soin de garantir que si nous modifions la valeur d'un nom de patient, par exemple, et que la ligne se répète, la nouvelle même valeur du nom soit attribuée aux deux lignes.
 - De plus, nous avons assuré la cohérence entre les fichiers. Ainsi, si un nom de patient apparaît à la fois dans le fichier Patient et dans le fichier Pharmacie, le même nouveau nom sera présent dans les deux fichiers, naturellement.
 - En ce qui concerne les valeurs nulles, si une case est NULL, nous ne lui attribuons aucune valeur et la laissons telle quelle.
- Sauvegarde des DataFrames anonymisés dans des fichiers Excel distincts.

Les données anonymisées ont été sauvegardées dans des fichiers séparés.

Voici un exemple de code de l'une des fonctions d'anonymisation :

```
import pandas as pd
import random
import names

def anonymize_names(name, name_map):
    if pd.notnull(name): # Verifie si la valeur de la cellule n'est pas NULL
        if name not in name_map:
            name_map[name] = names.get_first_name() + ' ' + names.get_last_name()
        return name_map[name]
    else:
        return name
```

L'anonymisation est une pratique essentielle qui doit être mise en œuvre dans tout projet qui implique la manipulation de données sensibles, et encore plus dans le domaine médical où la protection des données des patients est une priorité.

Nous nous engageons à maintenir ces normes éthiques élevées dans tous nos futurs projets de manipulation de données.

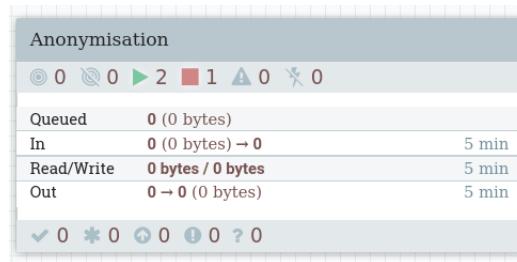


FIGURE 24 – Process Group de l'anonymisation



FIGURE 25 – Processeurs utilisés pour l'anonymisation

4.1.2 Première étape : "EXTRACT"

Dans la phase d'extraction de notre projet, nous avons traité plusieurs fichiers sources contenant des données variées et pertinentes pour notre domaine d'étude. Ces fichiers sources proviennent de différentes sources et présentent diverses structures et formats, ce qui nécessite une attention particulière lors de l'extraction des informations.

Au cours de cette étape, nous analysons et examinons chaque fichier source individuellement pour en comprendre la structure, les types de données et les relations entre les champs. Cette compréhension approfondie nous permet d'identifier les éléments clés nécessaires à l'extraction et de déterminer les transformations et les validations appropriées à appliquer ultérieurement.

Dans les sections suivantes, nous détaillerons chacun de ces fichiers sources un par un, en mettant en évidence leur contenu, leur structure et leur importance pour notre projet.

i Fichier patient

Le fichier_patients contient des données sur les patients enregistrés dans notre système. Ces données sont organisées dans une table comprenant les colonnes suivantes :

- **PATIENTID** : Il s'agit d'un identifiant unique attribué à chaque patient, permettant de les distinguer les uns des autres.
- **MR_NO** : Ce champ représente un numéro de dossier médical associé à chaque patient, facilitant leur suivi et leur identification dans le système.
- **HEALTHID** : Cette colonne peut contenir un identifiant spécifique lié à la santé de chaque patient, permettant peut-être de regrouper des informations médicales connexes.
- **DATE_REGESITER** : Il s'agit de la date à laquelle le patient s'est enregistré ou a été enregistré dans notre système, fournissant une référence temporelle pour chaque entrée.
- **PATIENT_NAME** : Ce champ contient le nom du patient, avec le nom de famille précédant le prénom, permettant une identification claire de chaque individu.

- **BIRTHDATE** : Cette colonne indique la date de naissance du patient, fournissant des informations sur leur âge et leur période de vie.
- **GENDER** : Il s'agit du genre du patient, indiqué par la lettre F pour féminin ou M pour masculin, permettant de différencier les données démographiques.
- **NATIONALITY** : Ce champ spécifie la nationalité du patient, indiquant leur pays d'origine ou de citoyenneté.
- **NATIONALID** : Cette colonne peut contenir un identifiant national spécifique attribué à chaque patient, facilitant leur identification au niveau national.
- **MARITALSTATUS** : Cela représente l'état matrimonial du patient, indiquant s'ils sont célibataires, mariés, divorcés, etc.
- **DEATH_DATE** : Si applicable, cette colonne enregistre la date de décès du patient, fournissant des informations sur leur statut vital.

PATIENTID	MR_NO	HEALTHID	DATE_REGESITER	PATIENT_NAME	BIRTHDATE	GENDER	NATIONALITY	NATIONALID	MARITALSTATUS	DEATH_DATE
37	28404			Nom1 Prenom1	28/01/41	F	France	1015093485		
53	13-77-80			Nom2 Prenom2	03-11-1393	F	France	1056399247		
91	15-34-82			Nom3 Prenom3	22/05/54	F	France	1025596816		August 23,2019
112	18-29-85			Nom4 Prenom4	02/09/65	F	France	1048651846		
146	12364	30000102797962		Nom5 Prenom5	10/04/67	F	France	1023389800		
189	13-86-88			Nom6 Prenom6	17/12/32	F	France	1058391564		
284	23-69-90			Nom7 Prenom7	19/09/69	F	France	0		
303	27-83-94			Nom8 Prenom8	02/05/89	F	Iceland	1023665544		
309	31881	30000178101527		Nom9 Prenom9	14/11/80	F	France	1032972398		
319	50665			Nom10 Prenom10	mercredi, septembre 14. 1977	M	France	1050945243		

FIGURE 26 – Exemple fichier Patient

ii Fichier pharmacie

Le fichier "pharmacy" contient des informations sur les médicaments délivrés aux patients. Voici une description des données présentes dans ce fichier :

- **ENCOUNTER_ID** : Identifiant de l'épisode de soin.
- **PATIENT_ID** : Identifiant du patient.
- **MR_NO** : Numéro du dossier médical du patient.
- **ACTIVITY_ID** : Identifiant de l'activité ou du service fourni.
- **ACTIVITY_TYPE** : Type d'activité ou de service.
- **PRODUCT_CODE** : Code du produit associé à l'activité.
- **DESCRIPTION** : Description du médicamenteux(avec la dose).
- **QUANTITY** : Quantité du produit ou du service fourni.
- **TIME_ARRIVED** : Heure d'arrivée du patient.
- **LINE_ORDER_DATE** : Date de la commande ou de la prescription.
- **ORDER_STAFF** : Personnel ayant passé la commande ou prescrit le service.
- **STAFF_NAT_ID** : ID national du staff médical.
- **CLINIC** : Nom de la clinique où l'activité a été fournie.

La figure ci-dessous illustre un exemple de fichier "Pharmacie" avec des données d'exemple pour visualiser la structure et le contenu de ce fichier.

ENCOUNTERID	PATIENTID	MR_NO	ACTIVITYID	ACTIVITYTYPE	PRODUCTCODE	DESCRIPTION	QUANTITY	TIME_ARRIVED	LINE_ORDER_DATE	DERER_STAFF	STAFF_NAT_ID	CLINIC
9249246	6710833	3672726	2712980	PHA	1191	OMACILLIN CAP 500 MG (AMOXICILLIN)	56	2019-01-01 07:19:57	2019-01-01 10:04:51 00074775		2392465457	Hospital275
9249246	6710833	3672726	2712968	PHA	1372	LOSEC 20 MG CAPS (OMEPRAZOLE)	60	2019-01-01 07:19:57	2019-01-01 10:04:19 00074775		2392465457	Hospital275
9249246	6710833	3672726	2712985	PHA	1665	CLARIVA TAB 250 MG (CLARITHROMYCIN)	56	2019-01-01 07:19:57	2019-01-01 10:05:21 00074775		2392465457	Hospital275
7742623	9889649	9913836	2712144	PHA	1567	FEROMIN TAB 190 (FERROUS SULPHATE) MG	0	2019-01-01 07:45:00	2019-01-01 09:19:51 02708224		1040662429	Hospital1
7742623	9889649	9913836	2712149	PHA	1570	BEFOLVIT TAB 1 MG (FOLIC ACID)	0	2019-01-01 07:45:00	2019-01-01 09:20:22 02708224		1040662429	Hospital1
2935782	8018298	1796628	2712305	PHA	1191	OMACILLIN CAP 500 MG (AMOXICILLIN)	63	2019-01-01 08:04:41	2019-01-01 09:29:55 00083412		2019015193	Hospital2
2935782	8018298	1796628	2712312	PHA	3408	FEROFOL TAB (FERROUS SULPHATE 150 MG + FOLIC ACID 500 MICRO GR/	30	2019-01-01 08:04:41	2019-01-01 09:30:28 00083412		2019015193	Hospital2
2935782	8018298	1796628	2712437	PHA	1602	CLEXANE INI 40 MG (ENOXAPARIN)	30	2019-01-01 08:04:41	2019-01-01 09:33:56 00083412		2019015193	Hospital2
2935782	8018298	1796628	2712395	PHA	1454	MICONAZOLE ORAL GEL	1	2019-01-01 08:04:41	2019-01-01 09:33:14 00083412		2019015193	Hospital2
2935782	8018298	1796628	2712341	PHA	1715	CALCIUM CARBONATE TAB 600 MG	30	2019-01-01 08:04:41	2019-01-01 09:33:24 00083412		2019015193	Hospital2
7846628	7470000	8015207	2712978	PHA	1197	JUSPRINT TAB 81 MG (ACETYLSALICYLIC ACID)	0	2019-01-01 08:06:00	2019-01-01 10:04:46 00057925		2160702383	Hospital143
8068113	1776770	7637747	2712087	PHA	1114	BECCOVIT TABLET (VIT B1+B6+B12)	0	2019-01-01 08:12:40	2019-01-01 09:15:47 02709630		2165755709	Hospital259
8068113	1776770	7637747	2712079	PHA	1487	TRAJENTA TAB 5 MG (INAGLUTIPINE)	0	2019-01-01 08:12:40	2019-01-01 09:15:34 02709630		2165755709	Hospital259
7206200	9691022	2495977	2712092	PHA	1676	VITAMIN D3 TAB 10000 UNIT (CHOLECALCIFERO)	60000	2019-01-01 08:15:17	2019-01-01 09:16:00 00083412		2019015193	Hospital22
9927455	3568637	2503463	2712504	PHA	1567	FEROMIN TAB 190 (FERROUS SULPHATE) MG	30	2019-01-01 08:18:23	2019-01-01 09:37:26 02708224		1040662429	Hospital1
9927455	3568637	2503463	2777951	PHA	1570	BEFOLVIT TAB 1 MG (FOLIC ACID)	60	2019-01-01 08:18:23	2019-01-01 10:25:20 02708224		1040662429	Hospital1
9927455	3568637	2503463	2777938	PHA	1567	FEROMIN TAB 190 (FERROUS SULPHATE) MG	90	2019-01-01 08:18:23	2019-01-01 10:24:37 02708224		1040662429	Hospital1
9927455	3568637	2503463	2712496	PHA	1129	TABUVAN TAB 80 MG (VALSARTAN)	30	2019-01-01 08:18:23	2019-01-01 09:36:51 02708224		1040662429	Hospital1
9927455	3568637	2503463	2712533	PHA	1593	NEXUM TAB 20 MG (ESOMEPRÁZOLE)	30	2019-01-01 08:18:23	2019-01-01 09:38:29 02708224		1040662429	Hospital1
4112830	1410342	8654813	2712024	PHA	1297	RANTAG TAB 150 MG (RANITIDINE)	30	2019-01-01 08:18:39	2019-01-01 09:11:15 02707948		2019015193	Hospital85
4112830	1410342	8654813	2712015	PHA	1715	CALCIUM CARBONATE TAB 600 MG	30	2019-01-01 08:18:39	2019-01-01 09:09:43 02707948		2019015193	Hospital85
4112830	1410342	8654813	2712014	PHA	3408	FEROFOL TAB (FERROUS SULPHATE 150 MG + FOLIC ACID 500 MICRO GR/	30	2019-01-01 08:18:39	2019-01-01 09:37:37 02707948		2019015193	Hospital85
2175335	1055365	1237641	2712353	PHA	3407	VISANNE 2 MG TAB	30	2019-01-01 08:23:02	2019-01-01 09:32:01 00083412		2019015193	Hospital22
6815228	5323265	5259728	2713477	PHA	3408	FEROFOL TAB (FERROUS SULPHATE 150 MG + FOLIC ACID 500 MICRO GR/	0	2019-01-01 08:24:33	2019-01-01 10:31:01 00083412		2019015193	Hospital2
5038301	9000340	9312082	2712173	PHA	3408	FEROFOL TAB (FERROUS SULPHATE 150 MG + FOLIC ACID 500 MICRO GR/	30	2019-01-01 08:32:19	2019-01-01 09:21:29 02707948		2019015193	Hospital85
5038301	9000340	9312082	2712178	PHA	1715	CALCIUM CARBONATE TAB 600 MG	30	2019-01-01 08:32:19	2019-01-01 09:21:42 02707948		2160702383	Hospital85
9961279	9857013	5716767	2712275	PHA	1715	CALCIUM CARBONATE TAB 600 MG	30	2019-01-01 08:33:39	2019-01-01 09:27:38 00057925		2160702383	Hospital143
9961279	9857013	5716767	2712265	PHA	3408	FEROFOL TAB (FERROUS SULPHATE 150 MG + FOLIC ACID 500 MICRO GR/	30	2019-01-01 08:33:39	2019-01-01 09:26:39 00057925		2160702383	Hospital143
2352631	1433993	6993797	2712506	PHA	1487	TRAJENTA TAB 5 MG (INAGLUTIPINE)	0	2019-01-01 08:34:33	2019-01-01 09:37:28 001B0001		2019015193	Hospital259

FIGURE 27 – Exemple fichier pharmacie

Ces informations aident à suivre ce qui se passe lors des visites de patients dans une clinique. Elles montrent quel personnel a fourni quels services ou produits à quels patients, quand et en quelle quantité. Ces données peuvent aider les cliniques à mieux comprendre comment elles utilisent leurs ressources, comment améliorer les soins pour les patients, et à étudier l'efficacité des différents traitements.

iii Fichier radiologie

Le fichier "radiology" contient des informations sur les services de radiologie. Voici une description des données présentes dans ce fichier :

- **ENCOUNTERID** : L'identifiant de l'épisode de soin.
- **PATIENTID** : L'identifiant du patient associé à l'interaction.
- **MR_NO** : Le numéro du dossier médical du patient.
- **ACTIVITYID** : L'identifiant de l'activité ou du service.
- **ACTIVITYTYPE** : Le type d'activité ou de service.
- **PRODUCTCODE** : Le code du produit associé à l'activité.
- **DESCRIPTION** : La description de l'activité ou du service.
- **ACTIVITYCODE** : Le code de l'activité.
- **UNITS_ORDERED** : Le nombre d'unités commandées.
- **QUANTITY** : La quantité de produit ou de service fournie.
- **CLINICAL_DISCHARGE_DATE** : La date de sortie clinique.
- **PHYSICAL_DISCHARGE_DATE** : La date de sortie physique.
- **ORDERINGCLINICIANCODE** : Le code du clinicien ayant passé la commande.
- **LINE_ORDER_DATE** : La date de la commande ou de la prescription. La figure ci-dessous illustre un exemple de fichier "radiology" avec des données d'exemple pour visualiser la structure et le contenu de ce fichier.

ENCOUNTERID	PATIENTID	MR_NO	ACTIVITYID	ACTIVITYTYPE	PRODUCTCODE	DESCRIPTION	ACTIVITYCODE	UNITS_ORDERED	QUANTITY	CLINICAL_DISCHARGE_DATE	PHYSICAL_DISCHARGE_DATE	LINE_ORDER_DATE
6930990	4545716	3894928	3024457	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-02-12 17:00:32	2019-02-12 15:07:04	
1143049	2948817	4133616	3266513	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-03-18 12:52:51	2019-03-18 12:52:54	2019-03-16 21:12:14
7366845	1139106	4706034	3444776	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-04-07 15:44:38	2019-04-07 15:44:40	2019-04-04 14:08:03
4792494	6518108	2019590	3443841	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-04-08 20:52:11	2019-04-08 20:52:14	2019-04-04 13:16:55
3698176	5658391	6680595	2292655	XRY	O0US0024	US GUIDED BIOPSY THYROID		1	0	2019-06-11 01:39:44	2019-06-11 01:45:53	2018-11-04 15:31:48
5985827	6303340	3090042	4023220	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-06-19 14:08:48	2019-06-19 14:08:54	2019-06-18 13:27:07
4348974	7560398	3352732	4144809	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-07-02 12:57:13	2019-07-02 12:57:14	2019-07-02 12:53:33
6181884	3227691	5260541	4144800	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-07-02 12:58:49	2019-07-02 12:58:54	2019-07-02 12:50:36
8893935	1758941	3114132	4164578	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-07-06 08:44:28	2019-07-06 08:44:33	2019-07-04 11:45:12
1675403	5423704	1150064	4145429	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-07-11 14:23:36	2019-07-11 14:23:39	2019-07-02 13:46:40
8137726	4600628	8583900	4135624	XRY	O0CT0008	CT ANGIOGRAPHY HEAD W/+W/O C		1	0	2019-07-16 15:05:51	2019-07-16 15:06:00	2019-07-01 14:31:06
1745364	3788316	4216546	4346260	XRY	O0XR0023	XR CHEST (PA,LAT)		1	0	2019-07-30 17:00:12	2019-07-30 17:00:15	2019-07-24 09:08:06
2999681	9978585	6234251	4401006	XRY	O0US0001	US ABDOMEN COMPLETE		1	0	2019-08-04 22:05:26	2019-08-04 22:05:29	2019-08-30 11:02:36
5666353	3432560	4594169	4623892	XRY	O0XR0100	XR NASOPHARYNX ADENOIDS		1	0	2019-08-31 14:35:04	2019-08-31 14:35:06	2019-08-30 09:00:00
5361003	9254794	3566714	4628910	XRY	O0US0003	US ABDOMEN (SINGLE ORGAN)		1	0	2019-08-31 18:09:35	2019-08-31 18:09:37	2019-09-05 13:00:00
2237449	3393443	1918196	4720529	XRY	O0US0100	US NEONATAL BRAIN SCAN		1	0	2019-09-10 15:11:22	2019-09-10 15:11:44	2019-10-09 06:00:00
9292826	7326553	6068543	4613404	XRY	O0US0001	US ABDOMEN COMPLETE		1	0	2019-09-10 15:28:43	2019-09-10 15:28:46	2019-08-28 15:39:57
4532913	4856799	2032788	4730870	XRY	O0US0001	US ABDOMEN COMPLETE		1	0	2019-09-17 22:37:05	2019-09-17 22:37:08	2019-09-11 13:18:01
5212424	9278522	8656798	5035909	XRY	O0US0001	US ABDOMEN COMPLETE		1	0	2019-10-20 17:48:04	2019-10-20 17:48:06	2019-10-14 10:05:22
1476346	4331386	3874318	5272905	XRY	O0XR0023	XR CHEST (PA,LAT)		1	0	2019-11-07 11:39:48	2019-11-07 11:41:43	2019-11-07 04:48:25
1078957	3594773	6539331	5271313	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-11-07 14:21:15	2019-11-07 14:21:17	2019-11-06 22:40:43
1242634	4235931	8528420	5276456	XRY	O0XR0023	XR CHEST (PA,LAT)		1	0	2019-11-07 16:25:14	2019-11-07 16:25:16	2019-11-07 10:29:48
4830922	7915830	8330055	5281739	XRY	O0XR0022	XR CHEST (AP OR PA)		1	0	2019-11-07 23:22:13	2019-11-07 23:22:13	2019-11-07 20:15:25
9050970	1027087	7802136	5283682	XRY	O0XR0094	XR KUB		1	0	2019-11-08 10:55:48	2019-11-08 10:55:50	2019-11-08 06:49:53
7360494	6172013	4796469	5276413	XRY	O0MR0190	MRI NECK W/O CONTRAST		1	0	2019-11-08 16:40:24	2019-11-08 16:40:27	2019-11-07 10:27:45
7360494	6172013	4796469	5278285	XRY	O0MR0221	MRI SPINE CERVICAL W/O C		1	0	2019-11-08 16:40:24	2019-11-08 16:40:27	2019-11-07 11:55:44
6850169	4387184	2546499	5276924	XRY	O0MR0285	MRA HEAD AND NECK W/+W/O C		1	0	2019-11-08 17:43:49	2019-11-08 17:43:50	2019-11-07 10:51:51
7093824	7011420	1925076	5272515	XRY	O0XR0028	XR CHEST PORTABLE		1	0	2019-11-09 00:00:00	2019-11-09 00:00:00	2019-11-07 02:23:32
3515898	8104281	5877676	5287083	XRY	O0XR0028	XR CHEST PORTABLE		1	0	2019-11-10 05:59:17	2019-11-10 05:59:40	2019-11-09 03:09:14

FIGURE 28 – Exemple fichier radiologie

Ces informations permettent de suivre les différents services de radiologie effectués pour un patient donné, y compris les types d'examens réalisés, les dates et heures de consultation, ainsi que les spécialistes impliqués. Elles sont essentielles pour le suivi médical du patient et l'analyse des services de radiologie fournis. Grâce à ces données, il est possible d'évaluer l'utilisation des services de radiologie, de surveiller les résultats des examens et d'optimiser les protocoles de traitement. Elles jouent un rôle important dans l'amélioration des soins de santé, la prise de décisions cliniques éclairées et la recherche médicale dans le domaine de la radiologie. La figure ci-dessus illustre un exemple de fichier "radiology" avec des données d'exemple pour visualiser la structure et le contenu de ce fichier.

iv Fichier transfert

Ce fichier contient des informations sur les transferts de patients entre différents services hospitaliers. Voici une description des données présentes dans ce fichier :

- **REGISTER_ID** : Identifiant d'enregistrement de l'admission.
- **FILE_ID** : Identifiant du dossier du patient.
- **ENCOUNTERID** : Identifiant de l'épisode de soin.
- **CURRENT_WARD** : Service hospitalier actuel où le patient est admis.
- **TRNS_WARD** : Service hospitalier vers lequel le patient a été transféré.
- **CURRENT_BED** : Lit actuel occupé par le patient dans le service hospitalier actuel.
- **TRANS_BED** : Lit vers lequel le patient a été transféré dans le service hospitalier de destination.
- **START_DATE** : Date et heure de début de l'admission ou du transfert.
- **END_DATE** : Date et heure de fin de l'admission ou du transfert.

Ces informations permettent de suivre les transferts de patients entre les services hospitaliers, en enregistrant les détails tels que les identifiants, les services actuels et de destination, les lits occupés, ainsi que les dates et heures de début et de fin des admissions ou des transferts. Ces données sont essentielles pour assurer un suivi précis des mouvements des patients dans l'hôpital, faciliter la coordination des soins et optimiser l'utilisation des ressources hospitalières. Elles peuvent également être utilisées pour analyser les schémas d'admission, évaluer l'efficacité des transferts et améliorer la planification des services hospitaliers. La figure ci-dessous illustre un exemple de fichier "transfert" avec des données d'exemple pour visualiser la structure et le contenu de ce fichier.

A	B	C	D	E	F	G	H	I
REGISTER_ID	FILE_ID	ENCOUNTERID	CURRENT_WARD	TRNS_WARD	CURRENT_BED	TRANS_BED	START_DATE	END_DATE
912774	835711	6473582	CARDIAC CARE UNIT (CCU)	407 ISOLATION			2019-07-06 03:03:22	2019-07-06 11:59:24
55070	56648	6376686	LONGE TERM CARE LONGE TERM (CCU)	406 ISOLATION	401-A	2018-11-08 10:59:00	2019-04-10 09:41:34	
449978	449978	4665219	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-03-08 04:56:38	2019-03-11 18:38:33	
428951	437832	3677079	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-07-19 12:03:12	2019-07-22 11:09:27	
914248	836311	4555929	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-07-12 04:23:43	2019-07-15 08:09:15	
916523	837690	6581649	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-07-28 02:29:52	2019-07-31 18:11:55	
912601	835661	2646566	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-07-05 01:56:36	2019-07-08 08:21:48	
416843	425727	3980244	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-07-15 12:27:55	2019-07-18 12:28:09	
127765	133231	1057659	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-07-09 04:58:48	2019-07-11 14:27:41	
249095	258075	5519204	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-07-18 16:01:46	2019-07-19 11:30:41	
916763	837377	7725049	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-07-24 03:12:17	2019-07-28 02:08:16	
520433	589379	9265723	CARDIAC CARE UNI CARDIAC CARE	BED NO (10)	BED NO (10)	2020-09-27 17:58:15	2020-09-28 02:21:00	
194913	202339	2023056	CARDIAC CARE UNIT (CCU)	BED NO (10)		2020-10-13 00:50:38	2020-10-22 13:36:40	
520433	589379	9265723	CARDIAC CARE UNI CARDIAC CARE	BED NO (10)	BED NO (10)	2020-09-28 02:21:57	2020-10-12 23:50:46	
743386	734551	5063556	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-12-24 00:33:42	2019-12-29 17:24:19	
932430	849371	3230773	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-12-12 23:33:44	2019-12-14 00:43:11	
163739	169219	4736901	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-12-14 01:07:47	2019-12-16 09:15:32	
464420	464420	3216066	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-12-19 15:44:06	2019-12-20 01:34:26	
31025	32140	9324406	CARDIAC CARE UNI CARDIAC CARE	BED NO (10)	BED NO.(12) ISC	2019-12-09 09:21:00	2019-12-12 17:49:55	
69281	74221	3079275	CARDIAC CARE UNI CARDIAC CARE	BED NO (10)	BED NO.(12) ISC	2019-12-16 09:16:00	2019-12-16 19:05:08	
279350	288315	2526429	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-12-20 01:34:59	2019-12-22 16:47:02	
946296	848894	4221487	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-12-08 05:22:19	2019-12-09 04:22:54	
949960	850621	8025260	CARDIAC CARE UNI ED-CPR	BED NO (10)	CPR - A	2019-12-30 01:01:00	2020-01-02 12:41:01	
690921	682087	9528634	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-12-22 23:36:37	2019-12-23 18:30:37	
301688	310639	7351582	CARDIAC CARE UNIT (CCU)	BED NO (10)		2020-01-02 20:52:31	2020-01-05 11:06:20	
890638	839021	3156691	CARDIAC CARE UNIT (CCU)	BED NO (10)		2019-08-15 03:58:23	2019-08-18 14:39:59	

FIGURE 29 – Exemple fichier transfert

v Fichier Laboratoire

Le fichier "laboratoire" contient des informations sur les tests et analyses effectués en laboratoire. Voici une description des données présentes dans ce fichier :

- **ENCOUNTERID** : Identifiant de l'épisode de soin.
- **PATIENTID** : Identifiant du patient associé à l'interaction.
- **MR_NO** : Numéro du dossier médical du patient.
- **ACTIVITYID** : Identifiant de l'activité ou du service.
- **ACTIVITYTYPE** : Type d'activité ou de service.
- **PRODUCTCODE** : Code du produit associé à l'activité.
- **DESCRIPTION** : Description de l'activité ou du service.
- **ACTIVITYCODE** : Code de l'activité.
- **UNITS_ORDERED** : Le nombre d'unités commandées.
- **QUANTITY** : La quantité de produit ou de service fournie.
- **CLINICAL_DISCHARGE_DATE** : Date de sortie clinique.
- **PHYSICAL_DISCHARGE_DATE** : Date de sortie physique.
- **ORDERINGCLINICIANCODE** : Code du clinicien ayant passé la commande. La figure ci-dessus illustre un exemple de fichier "laboratoire" avec des données d'exemple pour visualiser la structure et le contenu de ce fichier.
- **LINE_ORDER_DATE** : Date de la commande ou de la prescription.

Ces informations permettent de suivre les tests et analyses effectués en laboratoire pour un patient donné. Elles incluent des détails tels que les identifiants d'interaction, les identifiants de patient, les numéros de dossier médical, les codes et descriptions des tests, les quantités commandées et fournies, les dates de sortie clinique et physique, ainsi que les informations sur le clinicien ayant passé la commande. Ces données sont essentielles pour le suivi des résultats des tests, l'évaluation de la santé du patient, la coordination des soins médicaux et l'analyse des schémas d'utilisation des services de laboratoire. Elles jouent un rôle important dans la prise de décisions cliniques, la gestion des dossiers médicaux et l'amélioration de la qualité des soins.

ENCOUNTERID	PATIENTID	MR_NO	ACTIVITYID	ACTIVITYTYPE	PRODUCTCODE	DESCRIPTION	ACTIVITYCODE	UNITS_ORDERED	QUANTITY	NICAL_DISCHARGE_DATE	SICAL_DISCHARGE_DATE	ORDERINGCLINICIANCODE	LINE_ORDER_DATE
4741405	7801989	4131851	2635849	LAB	1853	CARDIAC PROFILE (CK,LDH,SGOT		1	1	#####	#####		2018-12-12 00:26:30
4741405	7801989	4131851	2635848	LAB	2395	AMYLASE		1	1	#####	#####		2018-12-21 00:26:30
4741405	7801989	4131851	2601541	LAB	2389	SGPT (ALT)		1	1	#####	#####		2018-12-17 00:24
4741405	7801989	4131851	2601538	LAB	2350	KFT (KIDNEY FUNCTION TEST)		1	1	#####	#####		2018-12-17 00:24
4741405	7801989	4131851	2647744	LAB	1939	ACID-FAST BACILLI STAIN		1	1	#####	#####		2018-12-23 12:01:27
4741405	7801989	4131851	2638615	LAB	1853	CARDIAC PROFILE (CK,LDH,SGOT		1	1	#####	#####		2018-12-22 00:12:41
4741405	7801989	4131851	2690608	LAB	2386	CK-MB		1	1	#####	#####		2018-12-29 00:03:27
4741405	7801989	4131851	2661759	LAB	2164	MAGNESIUM		1	1	#####	#####		2018-12-25 00:31:05
4741405	7801989	4131851	2567965	LAB	3257	CHEMISTRY ROUTINE (GLU,UREA)		1	1	#####	#####		2018-12-12 00:13:44
4741405	7801989	4131851	2618498	LAB	1853	CARDIAC PROFILE (CK,LDH,SGOT		1	1	#####	#####		2018-12-19 00:43:00
4741405	7801989	4131851	2618496	LAB	2395	AMYLASE		1	1	#####	#####		2018-12-19 00:43:00
4741405	7801989	4131851	2618279	LAB	2388	SGOT (AST)		1	1	#####	#####		2018-12-19 00:14:58
4741405	7801989	4131851	2591667	LAB	1853	CARDIAC PROFILE (CK,LDH,SGOT		1	1	#####	#####		2018-12-16 00:11:41
4741405	7801989	4131851	2627523	LAB	1853	CARDIAC PROFILE (CK,LDH,SGOT		1	1	#####	#####		2018-12-20 00:26:33
4741405	7801989	4131851	2627522	LAB	2095	CALCIUM		1	1	#####	#####		2018-12-20 00:26:33
4741405	7801989	4131851	2641916	LAB	2673	SPUTUM CULTURE		1	1	#####	#####		2018-12-23 01:00:07
4741405	7801989	4131851	2635847	LAB	2108	BILIRUBIN, DIRECT		1	1	#####	#####		2018-12-21 00:26:30
4741405	7801989	4131851	2638621	LAB	3181	COAGULATION PROFILE		1	1	#####	#####		2018-12-22 00:13:25
4741405	7801989	4131851	2567760	LAB	2386	CK-MB		1	1	#####	#####		2018-12-11 22:10:36
4741405	7801989	4131851	2585528	LAB	2386	CK-MB		1	1	#####	#####		2018-12-14 00:31:58
4741405	7801989	4131851	2610152	LAB	2388	SGOT (AST)		1	1	#####	#####		2018-12-18 00:36:07
4741405	7801989	4131851	2615245	LAB	2147	STOOL ANALYSIS		1	1	#####	#####		2018-12-18 12:43:44
4741405	7801989	4131851	2702421	LAB	2386	CK-MB		1	1	#####	#####		2018-12-31 01:10:52
4741405	7801989	4131851	2688084	LAB	3530	PROFILE DESCRIPTION - XDXIXFX		1	1	#####	#####		2018-12-28 00:11:19
4741405	7801989	4131851	2693178	LAB	3531	PROFILE DESCRIPTION - XCXBXC		1	1	#####	#####		2018-12-30 00:28:04
4741405	7801989	4131851	2693177	LAB	3530	PROFILE DESCRIPTION - XDXIXFX		1	1	#####	#####		2018-12-30 00:28:04
4741405	7801989	4131851	2577332	LAB	3243	CBC		1	1	#####	#####		2018-12-13 00:50:41
4741405	7801989	4131851	2567972	LAB	2096	PHOSPHOROUS		1	1	#####	#####		2018-12-12 00:13:44
4741405	7801989	4131851	2567964	LAB	1853	CARDIAC PROFILE (CK,LDH,SGOT		1	1	#####	#####		2018-12-12 00:13:44

FIGURE 30 – Exemple fichier laboratoire

vi Fichier Diagnosis

Le fichier "diagnosis" contient des informations sur les diagnostics médicaux des patients. Voici une description des données présentes dans ce fichier :

- **ENCOUNTERID** : Identifiant de l'interaction médicale.
- **PATIENTID** : Identifiant du patient associé à l'interaction.
- **MR_NO** : Numéro du dossier médical du patient.
- **DIAGNOSISTYPE** : Type de diagnostic.
- **DIAGNOSISCODE** : Code de diagnostic.
- **DESCRIPTION** : Description du diagnostic.
- **DIAGNOSIS_NOTE** : Note relative au diagnostic.
- **CONDITIONONSETFLAG** : Indicateur de début de la condition.
- **DATE_RECORDED** : Date d'enregistrement du diagnostic.
- **CLINICAL_DISCHARGE_DATE** : Date de sortie clinique.
- **PHYSICAL_DISCHARGE_DATE** : Date de sortie physique.

La figure ci-dessous illustre un exemple de fichier "diagnosis" avec des données d'exemple pour visualiser la structure et le contenu de ce fichier.

ENCOUNTERID	PATIENTID	MR_NO	DIAGNOSISTYPE	DIAGNOSISCODE	DESCRIPTION	DIAGNOSIS_NOTE	CONDITIONONSETFLAG	DATE_RECORDED	CLINICAL_DISCHARGE_DATE	PHYSICAL_DISCHARGE_DATE
6539203	2632677	7672297	P	S09.9	Unspecified injury of head			2018-12-31 00:00:00	2019-01-01 00:49:27	2019-01-01 00:49:31
6539203	2632677	7672297	P	U73.9	UNSPECIFIED ACTIVITY			2018-12-31 00:00:00	2019-01-01 00:49:27	2019-01-01 00:49:31
6539203	2632677	7672297	P	Y92.9	Unspecified place of occurrence			2018-12-31 00:00:00	2019-01-01 00:49:27	2019-01-01 00:49:31
6539203	2632677	7672297	P	W19	Unspecified fall			2018-12-31 00:00:00	2019-01-01 00:49:27	2019-01-01 00:49:31
1571732	6978981	1726844	P	K29.1	Other acute gastritis			2018-12-30 00:00:00	2019-01-01 04:44:13	2019-01-01 04:44:15
3423910	7877827	8079282	P	O99.8	Other specified diseases and conditions complicating pregnancy			2018-12-31 00:00:00	2019-01-01 03:58:15	2019-01-01 06:06:35
2671005	1480576	2107869	P	P03.4	Spontaneous abortion, incomplete, without complication			2018-12-30 00:00:00	2019-01-01 11:19:13	2019-01-01 11:19:18
2671005	1480576	2107869	P	O09.1	Duration of pregnancy 5-13 completed weeks			2018-12-30 00:00:00	2019-01-01 11:19:13	2019-01-01 11:19:18
7181276	9105336	3840384	P	U73.9	UNSPECIFIED ACTIVITY			2018-12-28 00:00:00	2019-01-01 11:57:20	2019-01-01 11:57:24
7181276	9105336	3840384	P	S32.02	Fracture of lumbar vertebra, L2 level			2018-12-28 00:00:00	2019-01-01 11:57:20	2019-01-01 11:57:24
7181276	9105336	3840384	P	W19	Unspecified fall			2018-12-28 00:00:00	2019-01-01 11:57:20	2019-01-01 11:57:24
7181276	9105336	3840384	P	Y92.9	Unspecified place of occurrence			2018-12-28 00:00:00	2019-01-01 11:57:20	2019-01-01 11:57:24
3666754	6900740	6011357	P	O60.0	PRETERM LABOUR WITHOUT DELIVERY			2018-12-30 00:00:00	2019-01-01 12:15:41	2019-01-01 12:15:44
5317366	2495200	4814206	P	O09.5	Duration of pregnancy 34-36 completed weeks			2018-12-28 00:00:00	2019-01-01 12:16:16	2019-01-01 12:16:19
5317366	2495200	4814206	P	O60	Preterm delivery			2018-12-28 00:00:00	2019-01-01 12:16:16	2019-01-01 12:16:19
5317366	2495200	4814206	P	O42	Premature rupture of membranes			2018-12-28 00:00:00	2019-01-01 12:16:16	2019-01-01 12:16:19
5317366	2495200	4814206	P	Z37.0	Single live birth			2018-12-28 00:00:00	2019-01-01 12:16:16	2019-01-01 12:16:19
6729682	6099628	3532515	P	I10	Essential (primary) hypertension			2018-12-28 00:00:00	2019-01-01 12:32:41	2019-01-01 12:32:46
6729682	6099628	3532515	P	I50.0	Congestive heart failure			2018-12-28 00:00:00	2019-01-01 12:32:41	2019-01-01 12:32:46
9813549	3994620	1272501	P	Z37.0	Single live birth			2018-12-28 00:00:00	2019-01-01 12:49:01	2019-01-01 12:49:03
9813549	3994620	1272501	P	O80	Single spontaneous delivery			2018-12-28 00:00:00	2019-01-01 12:49:01	2019-01-01 12:49:03
2450537	4160743	2417635	P	Z37.0	Single live birth			2018-12-31 00:00:00	2019-01-01 12:51:54	2019-01-01 12:51:56
2450537	4160743	2417635	P	O80	Single spontaneous delivery			2018-12-31 00:00:00	2019-01-01 12:51:54	2019-01-01 12:51:56
5930099	1489486	8608986	P	Z37.0	Single live birth			2018-12-29 00:00:00	2019-01-01 12:52:39	2019-01-01 12:52:41
5930099	1489486	8608986	P	O80	Single spontaneous delivery			2018-12-29 00:00:00	2019-01-01 12:52:39	2019-01-01 12:52:41
5261822	7475720	3293499	P	O80	Single spontaneous delivery			2018-12-30 00:00:00	2019-01-01 12:53:04	2019-01-01 12:53:06
5261822	7475720	3293499	P	Z37.0	Single live birth			2018-12-30 00:00:00	2019-01-01 12:53:04	2019-01-01 12:53:06
3759210	5093044	3787867	P	O09.1	Duration of pregnancy 5-13 completed weeks			2018-12-28 00:00:00	2019-01-01 13:05:14	2019-01-01 13:05:16
3759210	5093044	3787867	P	O02.1	Missed abortion			2018-12-28 00:00:00	2019-01-01 13:05:14	2019-01-01 13:05:16
5618977	8857439	2882818	P	O09.2	Duration of pregnancy 14-19 completed weeks			2018-12-31 00:00:00	2019-01-01 13:05:36	2019-01-01 13:05:39

FIGURE 31 – Exemple Diagnosis

Ces informations permettent de suivre les diagnostics médicaux des patients. Elles incluent des détails tels que les identifiants d'interaction, les identifiants de patient, les numéros de dossier médical, les types de diagnostic, les codes de diagnostic, les descriptions des diagnostics, les notes associées, les indicateurs de début de la condition, les dates d'enregistrement, ainsi que les dates de sortie clinique et physique. Ces données sont essentielles pour la gestion des dossiers médicaux, l'évaluation de la santé du patient, la planification des soins médicaux et l'analyse des schémas de diagnostic. Elles jouent un rôle crucial dans le processus de diagnostic, la prise de décisions cliniques et l'amélioration de la qualité des soins.

4.1.3 Deuxième étape : "TRANSFORM"

Dans cette étape, notre encadrant nous a fourni un fichier structuré en plusieurs branches pour simplifier le processus. La branche "Master" renseigne en détails chaque champ, incluant les types de fichiers, les formats et les règles de traitement à appliquer. La branche "Instructions" décrit les étapes à suivre pour l'intégration et le traitement des données, ainsi que des commentaires et des notes pour faciliter la compréhension du processus. La branche "Règles" énumère les diverses règles de validation et de transformation à mettre en œuvre lors du nettoyage et de la modification des données. Enfin, la branche "Mapping" présente le mapping des champs entre les sources de données et la base de données cible, assurant une intégration harmonieuse des données.

Champs	Type de fichier	Type / Taille / Format	Règle de traitement 1	Description de la règle de traitement 1
DateOfBirth	Patient	HeureDate(YYYY-MM-JJ HH:MM:SS)	V-Today-1	<= TODAY
FathersName	Patient	Varchar(100)	V-length100	Length<=100
FathersPreName	Patient	Varchar(100)	V-length100	Length<=100
Hospital	Patient	Varchar(100)	V-NotNull-1	<> NULL / blank (Rejet)
PlaceOfBirth	Patient	Varchar(100)	V-length100	Length<=100
SourcePatientNumber	Patient	Varchar(50)	V-length50	Length<=50
FileDateCreation	Patient	HeureDate(YYYY-MM-JJ HH:MM:SS)	D-DateCreation	Date et heure quand le Output file a été créé
PatientDeceased	Patient	Varchar(50)	D-PatientDeceased	If null, If [DateOfDeath] is a valid date, default to 'Oui'
DateOfDeath	Patient	HeureDate(YYYY-MM-JJ HH:MM:SS)	V-DateofDeath	>= [DateofBirth]

FIGURE 32 – Exemple branche Master

Type de règle	Logique des règles	Explication des règles	Action	Message	Status (flag in use or not)	Date modified
Validation	O-valeur par défaut seulement si le champ est rempli par "NULL", V-valider seulement si le champ n'est pas rempli par "NULL"	Il y a un rejet si la valeur n'est pas plus récente qu'il y a 125 ans	Rejet	Patient older than 125 years		
Validation	not > 125 years ago	Il y a un rejet si la valeur n'est pas plus petite ou égale à aujourd'hui	Rejet	Value must be less than or equal to today		
Validation	Standard date format validation (YYYY-MM-JJ HH:MM:SS)	Il y a un rejet si la valeur n'a pas un format de date standard 'YYYY-MM-JJ HH:MM:SS'	Rejet	Invalid date format. Must be YYYY-MM-DD HH:MM:SS		
Validation	Alpha Characters only	Il y a un avertissement si la valeur ne représente pas des lettres seulement	Avertissement	Alpha characters only		
Validation	Alpha Characters only	Il y a un rejet si la valeur ne représente pas des lettres seulement	Rejet	Alpha characters only		
Validation	<> NULL / blank (Rejet)	Il y a un rejet si le champ est rempli par "NULL" ou s'il n'y a pas une valeur	Rejet	Valeur ne peut pas être null ou vide		
Validation	<> NULL / blank (Avertissement)	Il y a un avertissement si le champ est rempli par "NULL" ou s'il n'y a pas une valeur	Avertissement	La valeur est null ou vide		
Transform	Remove Leading Zero	La valeur est transformée pour retirer les zéros au début	Aucun			
Default	If Blanc ==>Null	Si la valeur est Blanc, on met "Null"	Aucun			
Validation	Length <=100	La longueur ne doit pas dépasser 100, il y a une avertissement si longueur dépasse 100	Avertissement	La longueur ne doit pas dépasser 100 caractères		
Validation	Length <=50	La longueur ne doit pas dépasser 50, il y a une avertissement si longueur dépasse 50	Avertissement	La longueur ne doit pas dépasser 50 caractères		
Default	Date et heure quand le Output file a été créé	mettre la date et heure quand le Output file a été créé	Aucun			
Validation	>= [DateofBirth]	Il y a un rejet si la valeur n'est pas plus grande ou égale au champ [DateofBirth]	Rejet	Date must be greater than or equal to DateofBirth		
Default	If null, If [DateOfDeath] is a valid date, default to 'Oui'	La valeur représentée par défaut est "Oui" si la valeur du champ [DateOfDeath] est une date valide (si le champ est rempli par "NULL", sinon la valeur originale est conservée)	Aucun			

FIGURE 33 – Exemple branche Règles

Instructions	Commentaires
1 Valider que le fichier appartient à son domaine d'affaire (Patient, Service, transfer,...)	
2 Analyser les fichiers afin de définir les règles à appliquer et le mapping nécessaire	
3 Faire le Mapping des champs	
Nettoyage des données(- supprimer :les codes ASCII, les Caractères spéciaux, les vides,...	*Faites attention aux ponctuations (virgule, point virgule,...)
4 - Enelever les duplcats)	*Appliquer la bonne logique de suppression des duplcats selon le type de fichier
5 Appliquer les règles	
Générer une fichier "Rapport de validation" Contenant les détails suivants:	
- Nombre de records initiaux (Données source) avec un %	
- Nombre des records rejetés et les règles appliquées avec un %	
- Nombre des records avec avertissement et les règles appliquées avec un %	
6 - Aperçu des records concernés par les rejets et les avertissements	
7 Appliquer un processus d'assurance qualité	

FIGURE 34 – Exemple branche Instructions

Patient		
Output file		
PatientNumber		
Hospital	Hospital 1	Par défaut
DateOfBirth		
Gender		
PatientDeceased		
Datedeath		
PlaceOfBirth		
EthnicOrigin		
Nationality		
LastName		
FirstName		
Title		
MothersName		
MothersPreName		
FathersName		
FathersPreName		
FamilyDoctor		
NationalID		
FileDateCreation		

FIGURE 35 – Exemple branche Mapping

i Process Group

En Apache NiFi, un Process Group (groupe de processus) est une entité organisatrice et modulaire utilisée pour regrouper et organiser des composants de flux de données. Il offre un moyen de créer une structure hiérarchique pour gérer des flux de données complexes et faciliter la gestion, la réutilisation et la modularité des processus.

Un Process Group peut être considéré comme un conteneur logique qui peut inclure des composants tels que des processeurs, des connexions, des contrôleurs de flux, des groupes de ports, des variables de processus, des tâches planifiées, etc. Il permet de structurer et de hiérarchiser les flux de données en les organisant en sous-groupes, ce qui facilite la maintenance et la gestion des flux de données. Les Process Groups peuvent être utilisés pour encapsuler des fonctionnalités spécifiques, des tâches ou des processus connexes au sein d'une architecture de flux de données plus large. Cela permet de gérer et de contrôler efficacement différents aspects du flux de données, tels que la collecte, la transformation, la validation, la routage, etc.

En utilisant la structure hiérarchique des Process Groups, nous avons pu diviser notre workflow en sous-groupes logiques et fonctionnels. Cela permet une meilleure organisation et une compréhension plus intuitive de notre flux de données global. De plus, cela facilite également la réutilisation des Process Groups dans d'autres workflows similaires, améliorant ainsi l'efficacité de notre travail. Nous avons créé deux Process Groups pour améliorer la visualisation et la gestion de notre workflow. Le premier Process Group, que nous avons nommé "Mapping", est dédié au processus de mapping des données. Il regroupe les composants nécessaires pour cette étape, tels que les processeurs de transformation, les contrôleurs de flux et les connexions correspondantes. En utilisant ce Process Group, nous avons pu organiser de manière cohérente les étapes de mapping des données, ce qui facilite la compréhension et la maintenance de cette partie du workflow.

De même, nous avons créé un deuxième Process Group appelé "Application des règles", qui se concentre sur l'application des règles définies pour nos données. Ce Process Group contient les processeurs et les connexions nécessaires pour exécuter les règles et filtrer les données en conséquence. En regroupant ces composants dans un Process Group distinct, nous avons pu visualiser clairement l'étape d'application des règles et isoler cette partie spécifique de notre workflow. Nous avons établi une connexion entre les deux Process Groups afin de faciliter le flux de données entre eux. Pour ce faire, nous avons créé un port de sortie du côté du Process Group "Mapping" et un port d'entrée du côté du Process Group "Règles".

La connexion entre ces deux ports permet de transférer le flux de données généré lors du processus de mapping du Process Group "Mapping" vers le Process Group "Règles" pour l'application des règles correspondantes. Cette liaison permet d'établir une transition fluide et cohérente entre les différentes étapes de notre workflow.

Lorsque les données sont mappées dans le Process Group "Mapping", elles sont acheminées vers le port de sortie spécifié. Ensuite, ces données sont reçues par le port d'entrée du Process Group "Règles", où les règles définies sont appliquées pour effectuer les validations et les filtrages nécessaires.



FIGURE 36 – process groupe utilisé

Cette connexion entre les deux Process Groups facilite la transmission continue et automatisée des données entre les étapes de mapping et d'application des règles. Elle évite également les pertes ou les interruptions de données et garantit un flux de travail harmonieux.

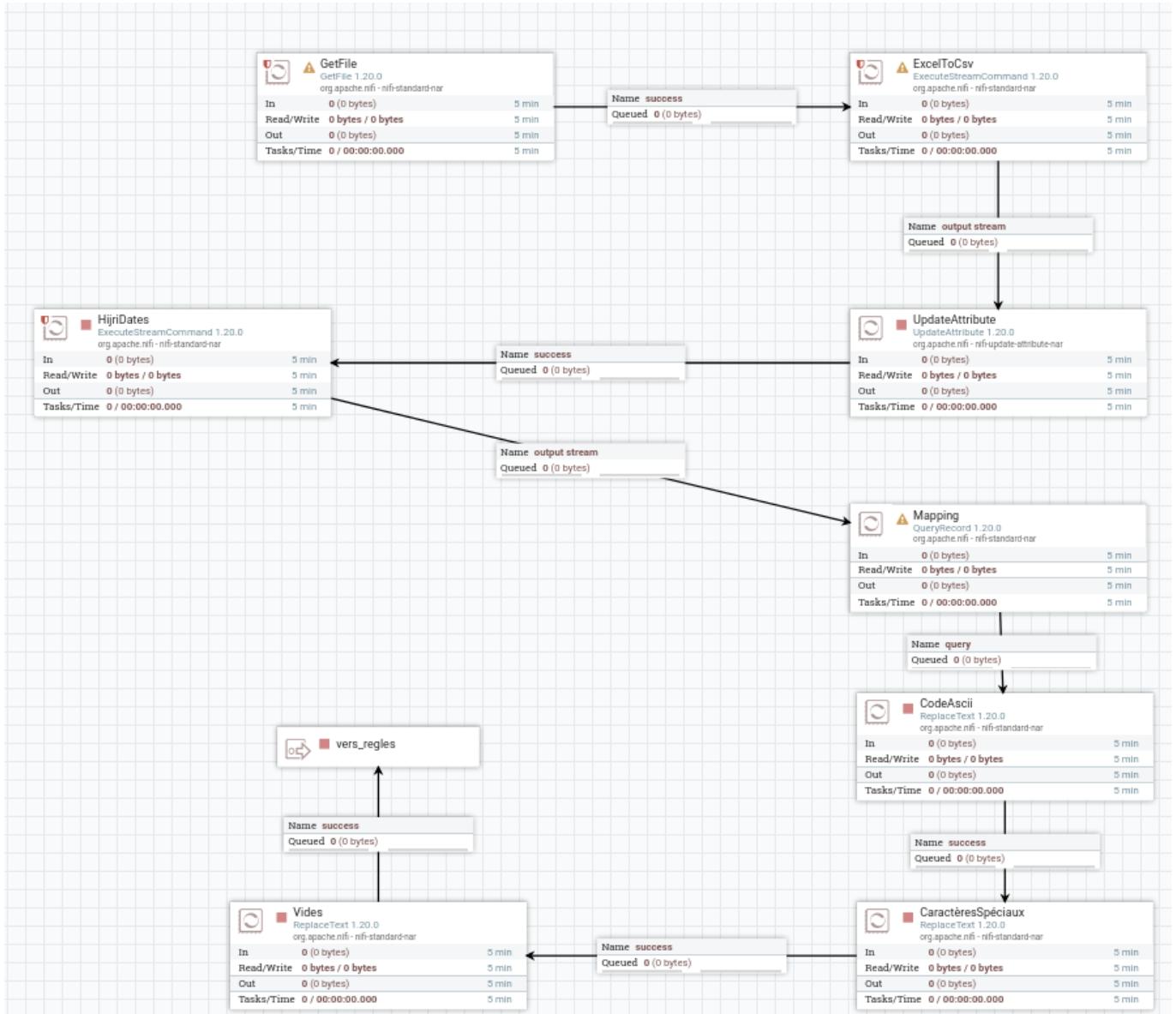


FIGURE 37 – Groupe "Mapping détaillé"

Ce workflow représente les différentes étapes du groupe mapping et illustre les interactions entre elles, Voici une explication détaillée de chaque étape :

ii Prétraitements

Voici une description des différentes étapes mentionnées dans les instructions pour le prétraitement des données :

- Valider que le fichier appartient à son domaine d'affaires : Avant de procéder à tout traitement, il est important de vérifier que le fichier de données concerne bien le domaine d'affaires spécifique pour lequel les règles d'intégration sont définies. Cela permet de s'assurer de l'applicabilité des règles et de l'adéquation des données au contexte.
- Analyser les fichiers afin de définir les règles à appliquer et le mapping nécessaire : Cette étape consiste à examiner attentivement les fichiers de données pour comprendre leur structure, les types de données qu'ils contiennent et les relations entre les différentes colonnes. L'objectif est de définir les règles d'intégration appropriées en fonction des besoins spécifiques du projet et de déterminer le mapping requis pour aligner les données entre différentes sources ou formats.
- Faire le mapping des champs : Une fois que les règles d'intégration ont été définies, il est nécessaire de faire correspondre chaque champ ou colonne du fichier source avec le champ correspondant dans le format de destination. Cela implique de créer un schéma de mapping qui associe les champs source et cible, en tenant compte des transformations ou des ajustements nécessaires.
- Nettoyage des données : Cette étape vise à nettoyer les données en appliquant différentes opérations, telles que la suppression des caractères ASCII non pertinents, des caractères spéciaux indésirables et des espaces vides. De plus, il est important de supprimer les doublons afin d'éliminer les entrées en double qui pourraient fausser les résultats. Il est également mentionné de faire attention aux ponctuations lors du nettoyage des données. Le prétraitement le plus complexe à mettre en place était la conversion des dates qui était dans un format hijri vu que les données proviennent d'un hôpital d'un pays du Golfe en un format grégorien. La conversion de ces dates vers le format grégorien était essentielle pour assurer la cohérence et la compatibilité avec les autres systèmes utilisés dans notre étude. Nous avons pu surmonter ce défi en utilisant une API spécifique appelée "ConvertToDate". Cette API a été spécialement conçue pour effectuer cette conversion en prenant les dates Hijri en entrée et en fournissant les dates correspondantes au format grégorien en sortie.

Mapping :

Le mapping des données consiste à associer des informations provenant de différentes sources ou formats aux champs spécifiques d'un système cible, garantissant ainsi une intégration cohérente et précise des données. Ce processus implique de définir des correspondances entre les éléments sources et les champs de destination, et peut inclure des conversions de formats, l'attribution de valeurs par défaut, la transformation des données ou toute autre action nécessaire pour aligner les informations avec les besoins du système cible. En d'autres termes, le mapping permet de relier les éléments des diverses sources de données aux champs appropriés du système dans lequel nous souhaitons intégrer ces données, assurant ainsi une harmonisation efficace entre les informations sources et les exigences du système cible.

Voici une liste non exhaustive des mappings effectués dans le cadre de notre projet (celui du fichier Patient) :

- PatientNumber : Ce champ correspond au numéro d'identification unique attribué à chaque patient.
- Hospital : Ce champ indique le nom de l'hôpital ou de la clinique en question.

- DateOfBirth : Ce champ enregistre la date de naissance de chaque patient.
- Gender : Ce champ représente le genre du patient, indiqué par des valeurs telles que "M" pour masculin ou "F" pour féminin.
- PatientDeceased : Ce champ indique si le patient est décédé ou non.
- DateofDeath : Si le patient est décédé, ce champ enregistre la date de décès.
- PlaceOfBirth : Ce champ enregistre le lieu de naissance du patient.
- EthnicOrigin : Ce champ représente l'origine ethnique du patient.
- Nationality : Ce champ indique la nationalité du patient.
- LastName : Ce champ enregistre le nom de famille du patient.
- FirstName : Ce champ enregistre le prénom du patient.
- Title : Ce champ représente le titre ou le préfixe associé au nom du patient.
- MothersName : Ce champ enregistre le nom de la mère du patient.
- MothersPreName : Ce champ enregistre le prénom de la mère du patient.
- FathersName : Ce champ enregistre le nom du père du patient.
- FathersPreName : Ce champ enregistre le prénom du père du patient.
- FamilyDoctor : Ce champ indique le médecin traitant du patient.
- NationalID : Ce champ représente le numéro d'identification national du patient.
- FileDateCreation : Ce champ enregistre la date de création du fichier.

iii Règles d'intégration

Les règles que nous avons appliquées sur nos données sont variées et permettent de garantir la qualité et la cohérence des informations traitées. Voici une explication des règles, classées par type de règle et avec une description de leur logique et des actions associées.

Règles de validation (V) : Ces règles vérifient si les données respectent certaines conditions et génèrent des avertissements ou des rejets selon le cas.

- V-DateOfBirth-1 : Rejette les enregistrements si la date de naissance remonte à plus de 125 ans.
- V-Today-1 : Rejette les enregistrements si la date est supérieure à la date d'aujourd'hui.
- V-FormatDate-1 : Rejette les enregistrements si la date n'est pas au format standard 'AAAA-MM-JJ HH:MM:SS'.
- V-Alpha-2 : Avertit si les données ne contiennent que des caractères alphabétiques.
- V-Alpha-1 : Rejette les enregistrements si les données ne contiennent pas uniquement des caractères alphabétiques.
- V-NotNull-1 : Rejette les enregistrements si le champ est NULL ou vide.
- V-NotNull-2 : Avertit si le champ est NULL ou vide.
- V-length100 : Avertit si la longueur du champ dépasse 100 caractères.
- V-length50 : Avertit si la longueur du champ dépasse 50 caractères.
- V-DateofDeath : Rejette les enregistrements si la date de décès est antérieure à la date de naissance.
- V-Num-1 (Validation, Chiffres uniquement) : Il y a un rejet si la valeur ne représente pas uniquement des chiffres. La valeur doit être numérique uniquement.
- V-Quantity-1 (Validation, Quantité > 0) : Il y a un rejet si la valeur n'est pas supérieure à 0. La valeur doit être strictement positive.

V-Alpha-1	Validation	Alpha Characters only	Il y a un rejet si la valeur ne représente pas des lettres seulement	Rejet
V-NotNull-1	Validation	<> NULL / blank (Rejet)	Il y a un rejet si le champ est rempli par "NULL" ou s'il n'y a pas une valeur	Rejet
V-NotNull-2	Validation	<> NULL / blank (Avertissement)	Il y a un avertissement si le champ est rempli par "NULL" ou s'il n'y a pas une valeur	Avertissement

FIGURE 38 – exemple règles de validation

Règles de transformation (T) : Ces règles modifient les données en fonction de certaines conditions.

- T-RemoveLeadingZero-1 : Supprime les zéros en début de champ.
- T-PatientNumber_1 (Transformation, Numéro de patient) : Le numéro de patient dans le fichier CSV final est concaténé avec le nom de l'hôpital. Le format est "Hopital-SourcePatientNumber".
- T-EncounterNumber-1 (Transformation, Numéro d'épisode) : Le numéro d'épisode dans le fichier CSV final est concaténé avec le nom de l'hôpital et le type d'épisode. Le format est "Hopital-EncouterType-EncouterNumber".
- T-BedNumber-1 (Transformation, Numéro de lit) : Le numéro de lit dans le fichier CSV final est concaténé avec le nom de l'hôpital, le Ward et le numéro de chambre. Le format est "Hopital-Ward-RoomNumber-BedNumber".
- T-RoomNumber_1 (Transformation, Numéro de chambre) : Le numéro de chambre dans le fichier CSV final est concaténé avec le nom de l'hôpital et le Ward. Le format est "Hopital-Ward-RoomNumber".
- T-RoundInteger-1 (Transformation, Arrondir à un nombre entier) : La valeur est transformée pour être arrondie à un nombre entier supérieur.

T-PatientNumber_1	Transformation	[Hospital] + "-" + [SourcePatientNumber]	rempli par "NULL", sinon la valeur originale est conservée Le numero de patient dans le fichier CSV final doit être concaténé avec le nom de l'hôpital: Hopital-SourcePatientNumber	Aucun
T-EncounterNumber-1	Transformation	[Hospital] + '-' + [EncounterType] + '-' + [EncounterNumber]	Le numero d'épisode dans le fichier CSV final doit être concaténé avec le nom de l'hôpital et le type d'épisode: Hopital-EncouterType-EncouterNumber	Aucun
T-BedNumber-1	Transformation	[Hospital] + '-' + [Ward] + '-' + [RoomNumber] + '-' + [BedNumber]	Le numero de lit dans le fichier CSV final doit être concaténé avec le nom de l'hôpital, le Ward et le numéro de chambre: Hopital-Ward-RoomNumber-BedNumber	Aucun

FIGURE 39 – exemple règles de transformation

Règles par défaut (D) : Ces règles attribuent des valeurs par défaut aux champs en fonction de certaines conditions.

- D-Null-1 : Remplace les valeurs vides par NULL.
- D-DateCreation : Insère la date et l'heure de création du fichier de sortie.
- D-PatientDeceased : Attribue la valeur "Oui" si le champ [DateOfDeath] est une date valide et si le champ est NULL, sinon conserve la valeur originale.
- D-BedNumber-1 (Défaut, Numéro de lit) : Si le champ [BedNumber] est nul ou non rempli, la valeur par défaut est [Hospital] + '-' + [Ward] + '-' + [RoomNumber] + '-' + [NULL]. Mettre NULL si le champ [BedNumber] est vide.
- D-RoomNumber_1 (Défaut, Numéro de chambre) : Si le champ [RoomNumber] est nul ou non rempli, la valeur par défaut est [Hospital] + '-' + [Ward] + '-' + [NULL]. Mettre NULL si le champ [RoomNumber] est vide.
- D-Sequence-1 (Défaut, Séquence) : En faisant un GROUPBY sur le IDPATIENT dans le fichier Diagnosis, et en se basant sur la date de prise en charge (DiagnosisDateTime), on classe les soins donnés à un patient par ordre croissant.
- D-Age-1 (Défaut, Âge) : Si le champ [StartTime] et le champ [DateOfBirth] sont tous deux remplis, la valeur par défaut représente la soustraction de la date de début de l'épisode et de la date de naissance, arrondie à l'année la plus proche. Si le champ [StartTime] ou le champ [DateOfBirth] est nul ou non rempli, la valeur originale est conservée.

D-Age-1	Default	If null, [StartTime] - [DateOfBirth], rounded up to nearest year	La valeur représentée par défaut est la soustraction de la date de début de l'épisode et de la date de naissance, arrondie à l'année la plus près (si le champ est rempli par "NULL", sinon la valeur originale est conservée)	Avertissement
---------	---------	--	--	---------------

FIGURE 40 – Exemple règles par défaut

Pour les appliquer, nous utilisons le processeur ExecuteStreamCommand qui permet de faire appel à un script sur la machine en précisant dans la propriété Command path le langage utilisé (Python dans notre cas) ainsi que le répertoire où le script se trouve et les arguments à faire passer à nos

méthodes.

On récupère dans notre script le flux de données entrant au processeur ExecuteStreamCommand en mettant la propriété **IgnoreSTDIN** à False, sous un format csv et donc on fait appel à `read_csv(sys.stdin)` de l'API **pandas** pour pouvoir faire le travail dessus.

On fait de même une fois qu'on a fini de travailler pour que le fichier résultant de notre travail soit dans le flux de sortie de notre processeur (`df.to_csv(sys.stdout, index=False)`).

En outre, nous avons codé une règle en plus qui dit que si jamais une colonne (un champs) ne contient que valeurs nulles, nous enlevons cette colonne.

Intégration avec OpenRefine et NiFi :

Comme expliqué précédemment, nous avons mis en place une fonctionnalité dans l'interface graphique permettant à l'utilisateur de charger un fichier dans différents formats, afin de pouvoir traiter les données de ce fichier.

Pour ce faire, nous avons utilisé l'interface pour effectuer l'importation initiale du fichier. Nous avons envoyé une requête HTTP depuis l'interface à l'aide de l'API Requests (avec la méthode `requests.post()`), puis utilisé le processeur "HandleHttpRequest" de NiFi pour récupérer cette requête. Ce processeur reste en écoute sur le numéro de port par lequel l'interface achemine la requête.

Ensuite, nous avons transformé le fichier reçu en un fichier CSV à l'aide d'un script Python exécuté par le processeur "ExecuteStreamCommand". Le format CSV est pratique pour l'utilisation de la bibliothèque Pandas. Nous avons également utilisé le processeur "ReplaceText" pour ajouter l'extension ".csv" au nom du fichier.

Une fois le fichier CSV obtenu, nous avons appliqué plusieurs prétraitements pour préparer les données en vue du mapping. Certains de ces prétraitements ont été réalisés à l'aide de scripts exécutés par les processeurs "ExecuteStreamCommand", tandis que d'autres ont été effectués avec le processeur "ReplaceText". Par exemple, nous avons utilisé des scripts pour remplacer plusieurs espaces par un seul espace, ainsi que pour supprimer les caractères ASCII non pertinents et les caractères spéciaux indésirables.

Une fois les prétraitements terminés, nous avons procédé au mapping des données à l'aide du processeur "ExecuteStreamCommand", en utilisant un dictionnaire récupéré à partir du fichier de configuration de NiFi (`nifi.properties`). Ce dictionnaire peut être modifié via l'interface graphique une fois que l'utilisateur a choisi son mapping. Il est structuré de la manière suivante : clé = Nom de la colonne source, valeur = Nom de la colonne cible (après mapping).

Voici un exemple du dictionnaire pour le fichier "Patient" :

```

1 dict_patient = {
2     'PatientNumber': 'MRN Number',
3     'DateOfBirth': 'DateOfBirth',
4     'Gender': 'Gender',
5     'Extra:PatientDeceased' : 'PatientDeceased',
6     'Extra:DateofDeath' : 'DateofDeath',
7     'Extra:PlaceOfBirth' : 'PlaceOfBirth',
8     'EthnicOrigin' : 'EthnicOrigin',
9     'Nationality' : 'Extra:Nationality',
10    'LastName' : 'LastName',
11    'FirstName' : 'FirstName',
12    'Title' : 'Title',
13    'Extra:MothersLastName' : 'MothersName',
14    'Extra:MothersFirstName' : 'MothersPreName',
15    'Extra:FathersLastName' : 'FathersName',
16    'Extra:FathersFirstName' : 'FathersPreName',
17    'Extra:FamilyDoctor' : 'FamilyDoctor',

```

```

18     'Extra:BloodRefusal' : 'BloodRefusal',
19     'Extra:OrganDonor' : 'OrganDonor',
20     'Extra:PrefLanguage' : 'PrefLanguage',
21     'Extra>LastUpdateDateTime' : 'LastUpdateDateTime',
22     'NationalIdentifier' : 'NationalID'}

```

On fait de même pour tout type de fichier, ces dictionnaires vont servir par la suite pour faire notre mapping grâce à la méthode mapping ci-dessous :

```

def mapping(df, choix, column_mapping, nom_fichier, type_fichier,
rejections_count):

    #Les mettre en majuscule
    column_mapping = {key.upper(): value for key, value in
    column_mapping.items()}

    if len(column_mapping) == 0:
        create_excel(df, len(df), 0, rejections_count)
    else :
        # Reorganiser les colonnes du DataFrame en suivant l'ordre
        # defini dans le dictionnaire
        df = df[column_mapping.values()]

    if choix == 1 : #Un fichier Patient
        # Iteration sur chaque colonne du dataframe
        for new_names, original_name in column_mapping.items():
            :
                # Verifier si la cle (le nom de colonne) est
                # presente dans le dictionnaire
            if original_name in df.columns:
                # Renommage de la colonne
                df.rename(columns={original_name: new_names},
                inplace=True)

        # Recherche du nom de l'hôpital dans le nom du fichier
        hopital_name = re.search('(Hop.*).csv', nom_fichier)

        # Creation de la colonne "Hospital" avec le nom de l'
        # hôpital
        df.insert(1, 'Hospital', hopital_name.group(1))

        # Ecriture du fichier CSV resultant sur la sortie
        # standard (stdout)
        df.to_csv(sys.stdout, sep=',', index=False)
    elif ...

```

Si nous remarquons dans l'interface graphique que le fichier d'entrée de l'utilisateur ne contient pas l'une des colonnes nécessaires, en l'occurrence une colonne "MandatoryField", nous renvoyons un dictionnaire vide (`len(dict) == 0`) et faisons appel au script qui nous génère le rapport de validation avec toutes les lignes rejetées (autrement dit, aucun travail n'est effectué). Sinon, nous renommons

chaque ancienne ligne par sa nouvelle valeur, en respectant l'ordre imposé par l'utilisateur. Nous remarquons dans le code ci-dessus qu'on a une colonne "Hospital" dont nous allons nous-mêmes chercher la valeur dans le nom du fichier.

Ce processus de mapping a permis d'associer les différentes colonnes du fichier CSV à des champs spécifiques dans notre système cible.

Ensuite, nous avons utilisé OpenRefine pour coder les règles de transformation des données. À l'aide de scripts en GREL (Google Refine Expression Language) et d'expressions régulières, nous avons appliqué nos règles.

Voici un exemple script GREL. Cette expression crée une nouvelle colonne "V-Alpha-2" qui affiche "avertissement" lorsque la valeur de la colonne "Nationality" ne correspond pas à des lettres de l'alphabet, et affiche une chaîne vide ("") dans les autres cas.

```
if ((( value . match( / ^[ a-zA-Z] +\$ / ) != null ) == false ) , " avertissement " ,  
" " )  
{
```

200 rows										Extensions		Wikibase				
Show as:	rows	records	Show:	5	10	25	50	100	500	1000	rows	< first	< previous	1	next	> last
All	PATIENTID	MRI_NO	HEALTHID	DATE_REGISTERED	PATIENT_NAME	BIRTHDATE	GENDER	NATIONALITY	NATIONALID	MARITALST						
1.	37	280404			Nonn1 Premon1	2801141	F	France	1011503485							
2.	53	132847			Nonn2 Premon2	09111993	F	France	1011503487							
3.	91	35-34-NO			Nonn3 Premon3	2305054	F	France	101150591616							
4.	112	18-29-95			Nonn4 Premon4	02050905	F	France	1014651386							
5.	146	12364	300000102797962		Nonn5 Premon5	1019487	F	France	1023889000							
6.	189	13-86-68			Nonn6 Premon6	1712102	F	France	1016391564							
7.	284	23-89-69			Nonn7 Premon7	18090909	F	France	0							
8.	303	27-83-94			Nonn8 Premon8	02050909	F	Iceland	1023695544							
9.	309	31181	30000178101527		Nonn9 Premon9	1411200	F	France	1032972998							
10.	5065	5065			Nonn10 Premon10	20060606	M	France	1050946243							
11.	322	38970			Nonn11 Premon11	09060606	F	France	1021044092							
12.	382	58109	30000144042367		Nonn12 Premon12	20060606	M	France	1050809002							
13.	417	62357			Nonn13 Premon13	samele juillet 10, 1965	M	France	1033500412							
14.	424	43577			Nonn14 Premon14	lundi, avril 15, 1984	F	France	1084397805							
15.	459	64486			Nonn15 Premon15	29/06/06	F	France	1011729874							
16.	560	560	30000010150149		Nonn16 Premon16	06/06/06	M	France	1002271290							
17.	576	62714	30000008576234		Nonn17 Premon17	19/12/94	F	France	1083405380							
18.	594	66416			Nonn18 Premon18	02/09/90	F	France	0							
19.	657	38256			Nonn19 Premon19	09/06/87	F	France	1021247844							
20.	670	49954	30000003810454		Nonn20 Premon20	24-1-1400	F	France	1021040427							
21.	679	32342	3000000732099025		Nonn21 Premon21	1984/10/01	F	France	1083371719							
22.	724	49244			Nonn22 Premon22	1970/08/01	F	France	1014719427							
23.	733	68781			Nonn23 Premon23	1984/06/19	F	France	1004427864							
24.	773	68781			Nonn24 Premon24	13/07/1945	F	France	1008003150							
25.	777	47828			Nonn25 Premon25	27/04/975	F	France	1012320547							
26.	805	61295			Nonn26 Premon26	26/03/1984	F	France	1							
27.	37	280404			Nonn27 Premon27	2803/19/41	F	France	1015093485							
28.	811	474691			Nonn28 Premon28	1984/07/15	F	France	1050946243							

FIGURE 41 – Fichier avant nettoyage

188 rows										Extensions		Wikibase				
Show as:	rows	records	Show:	5	10	25	50	100	500	1000	rows	< first	< previous	1	next	> last
All	PatientNumber	VLength80	Hospital	V-NotNull-1	DateOfBirth	V-DateOfBirth-1	V-Today-1	Gender	PatientDeceased	V-Alpha-2	DateOfI					
1.	37		Hospital 1	18411029	09/06/06			F								
2.	53		Hospital 1	19731128	09/06/06			F								
3.	91		Hospital 1	19544022	09/06/06			F								
4.	112		Hospital 1	18694042	09/06/06			F								
5.	146		Hospital 1	19874010	09/06/06			F								
6.	189		Hospital 1	19324017	09/06/06			F								
7.	284		Hospital 1	18894029	09/06/06			F								
8.	303		Hospital 1	19894002	09/06/06			F								
9.	309		Hospital 1	19804014	09/06/06			F								
10.	319		Hospital 1	18744024	09/06/06			M								
11.	322		Hospital 1	18664007	09/06/06			F								
12.	382		Hospital 1	19864018	09/06/06			F								
13.	417		Hospital 1	18654000	09/06/06			M								
14.	424		Hospital 1	18844015	09/06/06			F								
15.	459		Hospital 1	18864025	09/06/06			F								
16.	566		Hospital 1	18644000	09/06/06			M								
17.	575		Hospital 1	18844016	09/06/06			F								
18.	584		Hospital 1	18924022	09/06/06			F								
19.	607		Hospital 1	18674000	09/06/06			F								

FIGURE 42 – Fichier après nettoyage avec Open-Refine

Les colonnes ne contenant aucun champs avant ou après l'application des règles sont supprimées. Et les lignes rejetées à la phase de pré-traitements ou à la phase d'application des règles sont dirigées vers notre rapport de validation qui va les stocker et compter leur nombre en vue de pouvoir déterminer le pourcentage de lignes rejetées.

Passage à l'utilisation exclusive de NiFi :

Comme expliqué dans la comparaison d'outils, nous avons dû combiner deux outils mais pour que notre travail soit plus propre et optimal, nous avons décidé de tout faire sur NiFi.

On garde les étapes de prétraitements et de mapping et cette fois-ci on code les règles sur un script python que l'on appelle avec un processeur ExecuteStreamCommand.

Pour l'application de nos règles, nous avons créé une classe Python qui contient toutes nos fonctions. Chaque fonction correspond à une règle spécifique, et nous faisons appel à notre objet de classe chaque fois qu'une règle doit être appliquée.

En plus du dictionnaire écrit dans le fichier config qui concerne le mapping, l'utilisateur a le choix aussi de spécifier quelle règle s'applique sur quelle colonne, pour ce, nous ajoutons une propriété de type dictionnaire sur le même fichier après que l'utilisateur ait effectué son choix, les clés de ce dictionnaire correspondent aux colonnes de notre jeu de données, et les valeurs sont des listes de règles qui s'appliqueront à chaque colonne correspondante.

Notre script Python pourra ensuite accéder au fichier de configuration grâce au processeur ExecuteStreamCommand, et ainsi identifier les règles à appliquer à chaque colonne.

Nous itérerons ensuite sur le dictionnaire et vérifierons dans la liste des méthodes à appliquer sur chaque colonne si une règle valide existe. Si c'est le cas, nous l'appliquerons à la colonne correspondante.

Chaque règle est codée individuellement et prend en paramètre le nom de la colonne à laquelle elle s'applique. Nous passons également en paramètre une liste qui contient les lignes rejetées ou une liste des lignes avec avertissement, en fonction de la règle (rejet ou avertissement). Si nous trouvons des lignes qui ne respectent pas la règle, nous les supprimons des données en cas de rejet, et les ajoutons à la liste des lignes rejetées (ou averties). Nous utilisons également un dictionnaire pour enregistrer le nombre de lignes rejetées par règle, où la clé est la règle et la valeur est le nombre de données rejetées ou averties pour cette règle spécifique (pour chaque colonne). Ces informations sont essentielles pour la création du rapport de validation, notamment pour la méthode de génération du fichier Excel du rapport.

- Voici un exemple du code de 2 règles :

```
def V_length100(row, column_name, warnings_list, warnings_count):
    rule_name = 'V-length100'
    if len(str(row[column_name])) > 100:
        row_copy = row.copy()
        row_copy['Avertissement'] = rule_name
        warnings_list.append(row_copy)
        warnings_count[rule_name][column_name] += 1

def V_alpha1(row, column_name, reject_list, rejections_count):
    rule_name = 'V-alpha-1'
    if not str(row[column_name]).isalpha():
        row_copy = row.copy()
        row_copy['Rejet'] = rule_name
        reject_list.append(row_copy)
        rejections_count[rule_name][column_name] += 1
        return False
    return True
```

La première fonction, **V_length100**, vérifie si la longueur de la valeur dans une colonne donnée d'une ligne donnée est supérieure à 100 caractères. Si c'est le cas, la ligne est copiée avec un nouveau champ "Avertissement" contenant le nom de la règle, et cette copie est ajoutée à une liste de "warnings_list" (pour le rapport de validation). Le nombre de warnings pour cette règle est incrémenté dans le dictionnaire "warnings_count".

La deuxième fonction, **V_alpha1**, vérifie si la valeur dans une colonne donnée d'une ligne donnée ne contient que des caractères alphabétiques. Si la valeur n'est pas alphabétique, la ligne est copiée avec un nouveau champ "Rejet" contenant le nom de la règle, et cette copie est ajoutée à une liste de "reject_list". Le nombre de rejets pour cette règle est incrémenté dans le dictionnaire "rejections_count".

- Les listes des lignes dupliquées, rejetées et averties vont être concaténées dans un seul DataFrame, qui va nous servir et donc qui sera passé en paramètre avec le nombre de lignes initiales, ainsi que les dictionnaires des compteurs de rejets/avertissements par règle à la méthode de création du fichier Excel du rapport de validation.

Le flux de données résultant de notre processeur ExecuteStreamCommand va passer par le proces-

seur final "HandleHttpResponse" pour que le fichier final puisse être envoyé à l'interface comme réponse sur le même numéro de port, qui s'attend à une requête http (requests.get()).

4.1.4 Troisième étape : "LOAD"

Après les étapes de traitement des données, nous avons procédé à la phase de chargement où nous avons préparé les données pour leur utilisation ultérieure. Cette phase était dédiée à l'intégration des données transformées dans un format standard et facilement exploitable.

L'objectif principal lors du chargement était de produire un fichier de sortie dans différents formats, à savoir CSV, XLSX, JSON, YAML, Parquet, Avro et Apache ORC. Chacun de ces formats est non seulement récurrent, mais aussi crucial dans le domaine du big data.

Le format XLSX, spécifique à Microsoft Excel, permet un stockage de données plus complexe, y compris des formules et des données formatées, et est largement utilisé dans divers environnements professionnels. Le format CSV est universellement accepté et garantit la compatibilité avec un large éventail d'outils d'analyse et de visualisation de données.

JSON et YAML, quant à eux, sont des formats de données textuelles faciles à lire et à écrire pour les humains, tout en étant facilement générables et analysables par les machines. Ils sont fréquemment utilisés dans les échanges de données sur le web et pour la configuration des applications.

Parquet, Avro, et Apache ORC sont particulièrement efficaces dans le contexte du big data. Parquet est un format de fichier colonne open source pour Hadoop, offrant une performance de lecture des données supérieure, idéale pour les opérations d'analyse de données. Avro, utilisé pour la sérialisation des données dans Hadoop, est généralement associé à Kafka pour le traitement des données en temps réel, offrant une compacité et une vitesse appréciables. Apache ORC, également un format de stockage en colonnes pour Hadoop, est optimisé pour gérer de grands volumes de données, grâce à ses capacités avancées de compression des données.

En sélectionnant ces formats, nous avons veillé à ce que les données transformées soient facilement accessibles et utilisables par une multitude de systèmes, tout en reconnaissant leur importance fondamentale et leur récurrence dans le domaine du big data. Voici le fonction qui le permet :

```
def convert_CSV_to_chosen_type(source_file_path , target_file_path ,
target_file_type):
    # detect file type
    file_type = detect_file_type(source_file_path)
    # convert to data frame
    df = convert_to_DATAFRAME(source_file_path)
    if target_file_type == 1: # 1 csv
        convert_to_CSV(df , target_file_path)
    if target_file_type == 2: # 2 xlsx
        convert_to_XLSX(df , target_file_path)
    elif target_file_type == 3: # 3 json
        convert_to_JSON(df , target_file_path)
    elif target_file_type == 4: # 4 yaml
        convert_to_YAML(df , target_file_path)
    elif target_file_type == 5: # 5 parquet
        convert_to_parquet(df , target_file_path)
    elif target_file_type == 6: # 6 avro
        convert_to_avro(df , target_file_path)
    elif target_file_type == 7: # 7 APACHE orc
```

```

    convert_to_orc(df, target_file_path)
else:
    print("Invalid file type parameter.")

#L'une des methodes de conversion
def convert_to_CSV(df, target_file_path):
    # Convert to CSV and save
    output_path = target_file_path + ".csv"
    df.to_csv(output_path, index=False)

....
```

Nous avons également procédé à la création d'un rapport de validation sous la forme d'un document structuré, présentant les résultats détaillés des vérifications effectuées. Ce rapport permettait aux utilisateurs de comprendre rapidement les problèmes potentiels dans les données transformées, en mettant en évidence les erreurs ou les anomalies détectées.

4.1.4.1 Rapport de Validation

Un rapport de validation est un outil précieux pour évaluer la qualité et la conformité des données, en identifiant les problèmes potentiels et en aidant à prendre des mesures correctives. Il permet aux utilisateurs de visualiser et d'analyser rapidement les résultats de la validation, ce qui facilite la prise de décision et l'amélioration continue des processus de gestion des données. Il contient les éléments suivants :

- Statistiques globales : Il peut inclure des informations telles que le nombre total d'enregistrements traités, le nombre d'enregistrements valides, le nombre d'enregistrements rejetés ou comportant des erreurs, et le nombre d'avertissements.
- Erreurs et avertissements : Le rapport peut répertorier en détail les enregistrements qui ne respectent pas les règles de validation, en indiquant les champs concernés, la nature de l'erreur ou de l'avertissement, et éventuellement une description de la règle de validation enfreinte.

Nous avons utilisé Openpyxl pour pouvoir générer automatiquement un rapport de validation après application de nos règles.

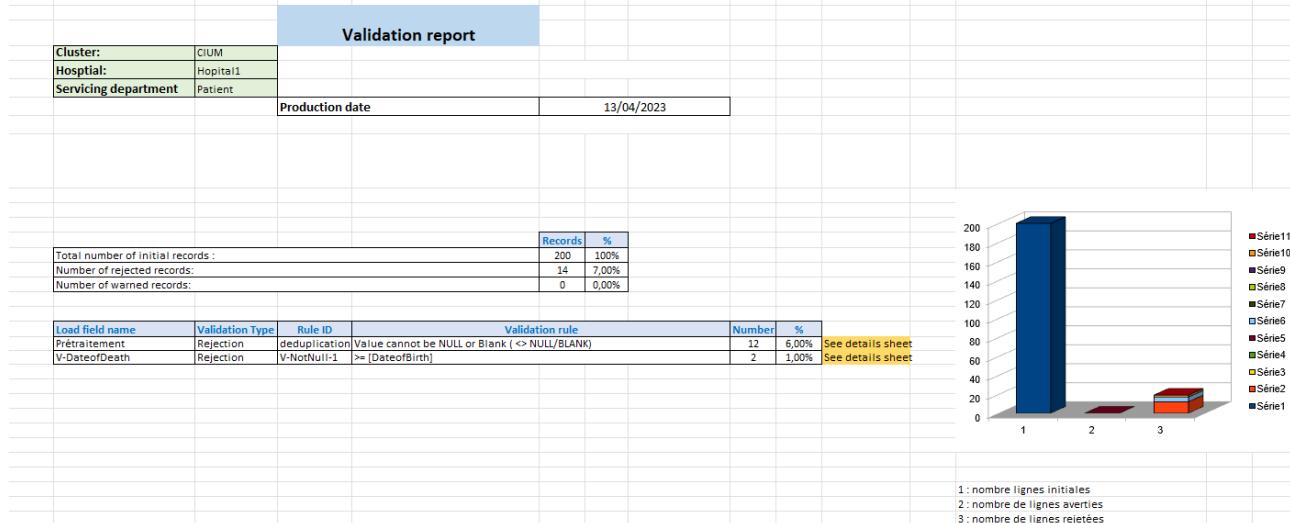
Cette API nous permet de modifier la mise en forme de nos feuilles excel y compris le format du texte, les couleurs du fond (remplissage), les bordures (et donc faire des tableaux), fusionner des cellules, changer les titres des feuilles, centrer le texte, changer les dimensions des cellules..etc.

Nous avons pu avoir un rapport de validation similaire à celui présenté par l'expert de données hospitalières.

Nous avons écrit un script pour la génération du fichier excel dynamiquement en se basant sur cette API qui prend en paramètres les lignes rejetées, les lignes avec avertissements (sous forme d'un DataFrame), le nombre de lignes initiales, un dictionnaire qui contient le nombre de lignes avec avertissement par règle et par colonne, et un autre dictionnaire qui contient le nombre de rejets par règle et par colonne aussi.

Nous créons un objet de type WorkBook() qui nous permet de créer le fichier Excel, nous avons 2 feuilles (worksheet) la première pour le résumé du travail effectué (Summary) et la deuxième qui se nomme Details et qui va contenir le dataframe des lignes rejetées et les lignes avec avertissement.

Dans la feuille Summary on trouve :



- Cluster : Cette section indique le cluster ou le groupe auquel les hôpitaux appartiennent (établissement d'hôpitaux).
- Hopital : Cette section précise le nom de l'hôpital associé aux données du rapport de validation. Cela peut indiquer l'origine ou la source des informations contenues dans le document.
- Servicing department : Cette section spécifie le service ou le département responsable de la gestion ou du traitement des données incluses dans le rapport de validation.
- Production date : Cette section indique la date et l'heure de production du rapport de validation. Dans cet exemple, la date de production est "13-04-2023".
- Records : Cette section présente des statistiques sur le nombre total de lignes dans le rapport de validation. Le nombre de lignes initiales va s'écrire dynamiquement avec le pourcentage dans la cellule correspondante dans le premier tableau bleu, pareil pour le nombre de lignes rejetées et avec avertissement.
- Total number of initial records : Il s'agit du nombre total de lignes initialement inclus dans le rapport de validation. Dans cet exemple, il y a 200 lignes au total.
- Number of rejected records : Cela indique le nombre de lignes qui ont subi un avertissement car elles n'ont pas satisfait les critères de validation.
- Number of warned records : Cette valeur indique le nombre de lignes qui ont subi un avertissement.

Le deuxième tableau sera consacré à la répartition des lignes rejetées et averties pour chaque règle et pour chaque colonne, accompagné du pourcentage correspondant à chaque valeur, ainsi que d'une explication détaillée de la règle et de son ID. Nous créerons une ligne pour chaque règle qui a été appliquée à nos données et qui a entraîné un rejet ou un avertissement, et nous n'en ajouterons pas pour celles qui ne répondent pas à ces critères.

Dans le cadre de notre analyse, nous souhaitons mettre en évidence les règles spécifiques qui

ont eu un impact sur les données, afin de fournir une compréhension approfondie de la qualité et de la validité des informations. Par conséquent, chaque règle qui a influencé les résultats sera minutieusement détaillée, en expliquant clairement son objectif, à quelle colonne elle a causé le rejet (avertissement) et en fournissant un identifiant unique.

- Load field name : Cette colonne spécifie le nom du champ associé aux règles qui ont causé un avertissement ou un rejet.
- Validation Type : Cette colonne indique le type de validation qui a été effectuée pour le champ spécifié. Dans cet exemple, il s'agit de 2 rejets (Rejection).
- Rule ID : Cette colonne fournit un identifiant ou un code unique pour la règle de validation appliquée. Cela permet d'identifier de manière spécifique la règle en question.
- Validation rule : Cette colonne décrit la règle de validation qui a été appliquée au champ spécifié. Dans cet exemple, la règle est "V-Not-Null-1".
- Number : Cette colonne indique le nombre de dossiers qui ont été affectés par la règle de validation spécifiée.
- % : Cette colonne représente le pourcentage du nombre total de dossiers affectés par la règle de validation spécifiée.

Et sur la partie "Details", nous retrouvons les lignes rejetées ou faisant objet d'un avertissement.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	PatientN	Hospital	DateOfBi	Gender	PatientDr	DateofDe	PlaceOfB	EthnicOri	Nationali	LastNam	FirstNam	Title	MothersF	MothersM	FathersN	FathersP	FamilyDo	Nationali	FileDate	C	Rejet
2	37	Hopital 1	1941-01-2 F					France	Nom27	Prenom27								1015093485		Duplication	
3	1527	Hopital 1	1987-10-2 M					France	Nom99	Prenom99								1048427866		Duplication	
4	4555	Hopital 1	1966-07-3 F					France	Nom111	Prenom1	MARRIED							1011551211		Duplication	
5	4563	Hopital 1	1991-01-2 M					France	Nom155	Prenom155								1073084930		Duplication	
6	4677	Hopital 1	1977-08-2 M					France	Nom156	Prenom156								1042010205		Duplication	
7	4883	Hopital 1	1948-11-C M					France	Nom157	Prenom157								1024573980		Duplication	
8	4919	Hopital 1	1991-11-2 M					France	Nom158	Prenom158								1074459817		Duplication	
9	4922	Hopital 1	1989-12-1 F					France	Nom159	Prenom159								1075745909		Duplication	
10	5008	Hopital 1	1985-12-C F					France	Nom160	Prenom160								1013991037		Duplication	
11	5031	Hopital 1	1988-09-1 F					France	Nom161	Prenom161								1059954857		Duplication	
12	6214	Hopital 1	1982-01-C F					France	Nom163	Prenom163								1034237576		Duplication	
13	6703	Hopital 1	1955-08-1 M					France	Nom167	Prenom167								1032602177		Duplication	
14	943	Hopital 1	1979-01-2 F				1978-01-09 00:00:00	France	Nom30	Prenom30								1014156671	V-DateofDeath		
15	2708	Hopital 1	1942-07-1 F				1941-09-12 00:00:00	France	Nom71	Prenom71								1057913889	V-DateofDeath		

Méthodes utilisées :

Nous avons utilisé la méthode `create_sheet()` pour créer les feuilles Excel nécessaires.

Ensuite, pour écrire les données fournies, nous avons utilisé la méthode `dataframe_to_rows()`, qui nous permet d'écrire ligne par ligne dans la feuille "Details"

Pour sélectionner une cellule spécifique et y inscrire les données, nous avons utilisé la méthode `worksheet.cell(indice de la ligne, indice de la colonne)`.

De plus, nous avons fusionné des cellules à l'aide de la méthode `worksheet.merge_cells()`, qui prend en compte les indices de début et de fin des lignes et des colonnes.

Pour des tâches plus avancées, telles que le centrage du texte, la modification du style du texte (gras, italique, etc.) et l'ajout de bordures autour des cellules, nous avons utilisé les méthodes `Alignment`, `Font` et `Border` respectivement.

Enfin, pour ajuster les dimensions d'une cellule, nous avons utilisé les méthodes `column_dimensions().width` et `column_dimensions().height`.

Nous avons utilisé la méthode `Workbook.save()` pour accomplir une tâche essentielle : sauvegarder notre Workbook, c'est-à-dire notre fichier Excel, dans le répertoire spécifié en paramètre. Cette opération permet de conserver toutes les modifications apportées aux feuilles, cellules et mises en forme, assurant ainsi la préservation de notre travail.

Graphique :

Nous avons utilisé la bibliothèque de création de graphiques "openpyxl" pour générer des graphiques dans notre feuille de calcul. Ce qui nous permet de mieux visualiser les nombre de lignes initiales,

nombre de lignes rejetées et averties par règle sous forme d'un histogramme.

Nous avons instancié un objet de la classe "BarChart" pour représenter notre graphique à barres. Cet objet est responsable de la configuration et de la personnalisation du graphique.

Nous avons défini le type de graphique en utilisant l'attribut "type" de l'objet "BarChart". Nous avons choisi un graphique à colonnes. De plus, nous avons défini le style du graphique en utilisant l'attribut "style". Dans notre cas, nous avons utilisé le style numéro 10.

Ensuite, nous avons ajouté un titre au graphique en utilisant l'attribut "title" de l'objet "BarChart". Nous avons également configuré les titres des axes y et x en utilisant les attributs "y_axis.title" et "x_axis.title" respectivement.

Ainsi, nous avons utilisé la classe "Reference" pour créer des références aux données du graphique dans notre feuille de calcul. Nous avons créé une référence aux données qu'on souhaite tracer. De plus, nous avons créé une référence aux libellés des règles de rejet voire d'avertissement.

Nous avons utilisé les méthodes "add_data" et "set_categories" de l'objet "BarChart" pour ajouter les données et les catégories au graphique. Nous avons passé les références de données et de libellés que nous avons créées précédemment en tant que paramètres de ces méthodes.

Enfin, nous avons utilisé la méthode "add_chart" de l'objet "worksheet" (représentant notre feuille de calcul) pour ajouter le graphique à une position spécifiée.

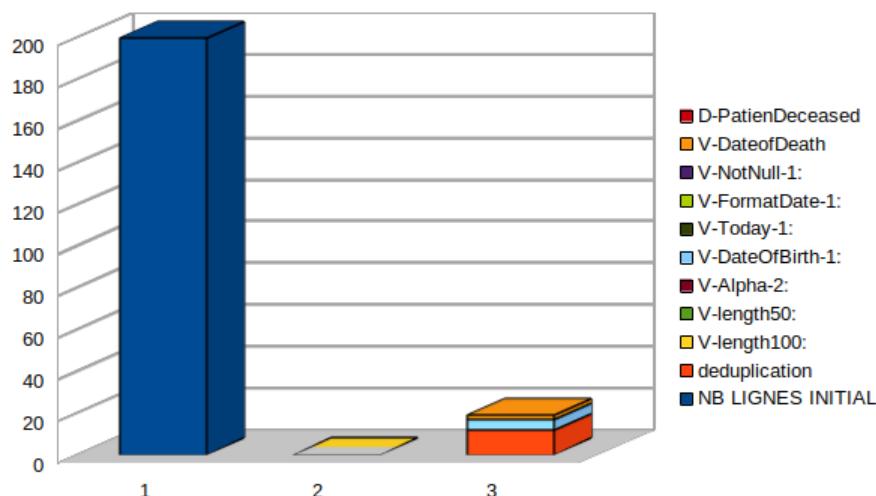


FIGURE 43 – Détails des nombres de lignes rejetées/averties par règle

Voici un fragment de code de la création dynamique du rapport de validation :

```
def create_excel(lines_df, initial_row_count, warnings_count,
rejections_count):
    rules_and_significations = {
        "V-Today-1": "<= TODAY",
        "V-DateOfBirth-1": "not > 125 years ago",
        "V-DateofDeath": ">= [DateofBirth]",
        "V-NotNull-1": "<> NULL / blank (Rejet)",
        "V-NotNull-2": "<> NULL / blank (Avertissement)",
        "V-length50": "Length <=50",
        "V-length100": "Length <=100",
        "V-alpha-1": "Alpha Characters only",
    }
```

```

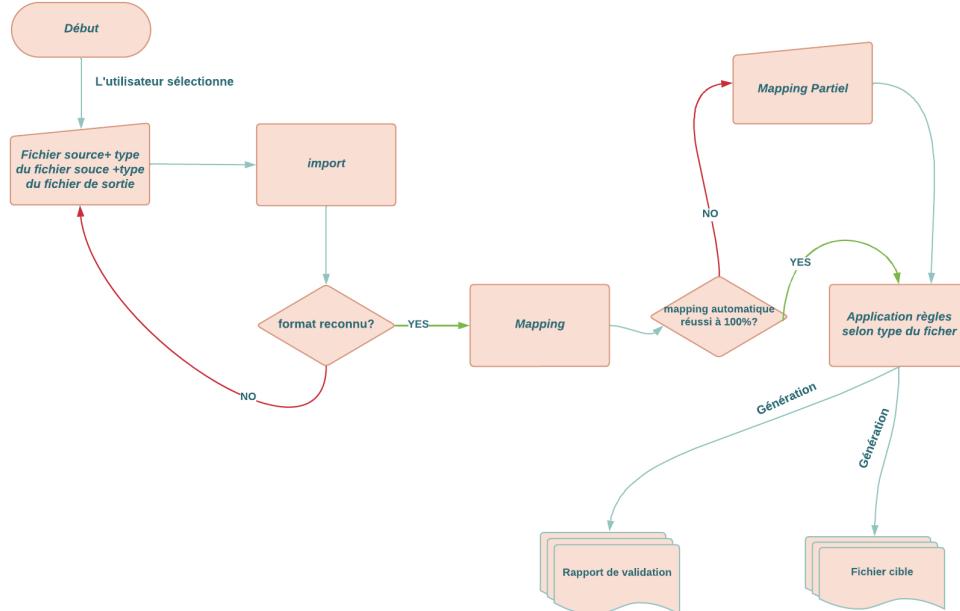
    "V-alpha-2" : "Alpha Characters only",
}
# calculer le total de lignes d'avertissements
total_warnings = sum(warnings_count.values())
# calculer le total de lignes rejetées
total_rejections = sum(rejections_count.values())
# Créer un nouveau classeur Excel
wb = openpyxl.Workbook()
worksheet = wb.active
worksheet.title = 'Summary'
# Merge cells D2:E3 and set the fill color to blue
cell_range = 'D2:E3'
worksheet.merge_cells(cell_range)
...

```

4.2 IHM et architecture logicielle pour l'interaction avec les composants du processus ETL

4.2.1 Conception de l'interface

La conception de l'interface est un élément essentiel de notre système. Il s'agit du point d'interaction principal entre l'utilisateur et notre application. Notre objectif est de créer une interface intuitive, conviviale et efficace, qui permet aux utilisateurs de naviguer facilement et d'accomplir leurs tâches avec un minimum d'effort.



Dans le schéma de conception ci-joint, nous avons représenté l'agencement des composants de l'interface, illustrant la façon dont un utilisateur interagirait avec notre système. Les éléments clés du schéma incluent l'importation des fichiers, le mapping et l'application des règles.

4.2.2 Implémentation de l'interface

Nous avons mis l'accent sur la simplicité lors de la conception de notre interface graphique, en utilisant des technologies appropriées telles que la bibliothèque **customTkinter** de Python. Nous avons veillé à ce que l'interface soit épurée et facile à comprendre pour les utilisateurs, en évitant de la surcharger avec des éléments superflus. De plus, nous avons appliqué les principes de conception actuels pour garantir une expérience utilisateur fluide et intuitive. Ainsi, les utilisateurs peuvent interagir facilement avec les fonctionnalités et les résultats de notre travail de recherche, sans se sentir dépassés par la complexité technique. L'interface offre une expérience conviviale et accessible à tous les niveaux de compétence.

L'interface graphique est composée de quatre *frames*, qui correspondent à des sections ou des écrans distincts.

Le premier *frame* correspond à l'étape d'authentification, qui joue un rôle fondamental dans la protection des données sensibles des utilisateurs et la sécurisation de leur expérience. En exigeant que les utilisateurs se connectent, nous garantissons un accès exclusif aux fonctionnalités de notre plateforme, renforçant ainsi la confidentialité et la sécurité des informations échangées. De plus, l'authentification est essentielle pour prévenir les activités abusives ou frauduleuses, protégeant ainsi l'intégrité de notre service.

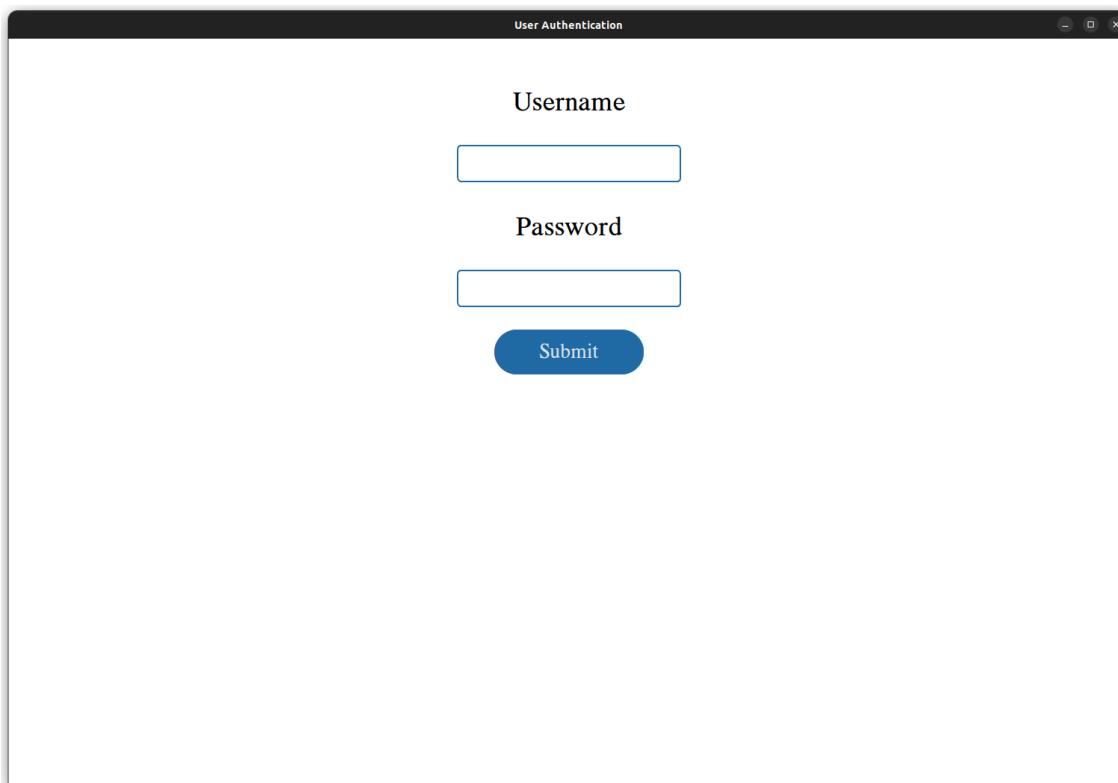


FIGURE 44 – écran d'authentification

Le deuxième *frame* permet à l'utilisateur d'importer un fichier et de spécifier son type parmi une liste prédéfinie comprenant des catégories telles que :

- Patient

- Encounter
- Transfer
- Diagnosis
- Procedure
- Service

Cette étape est importante pour permettre à l'application de proposer un mapping approprié des données.

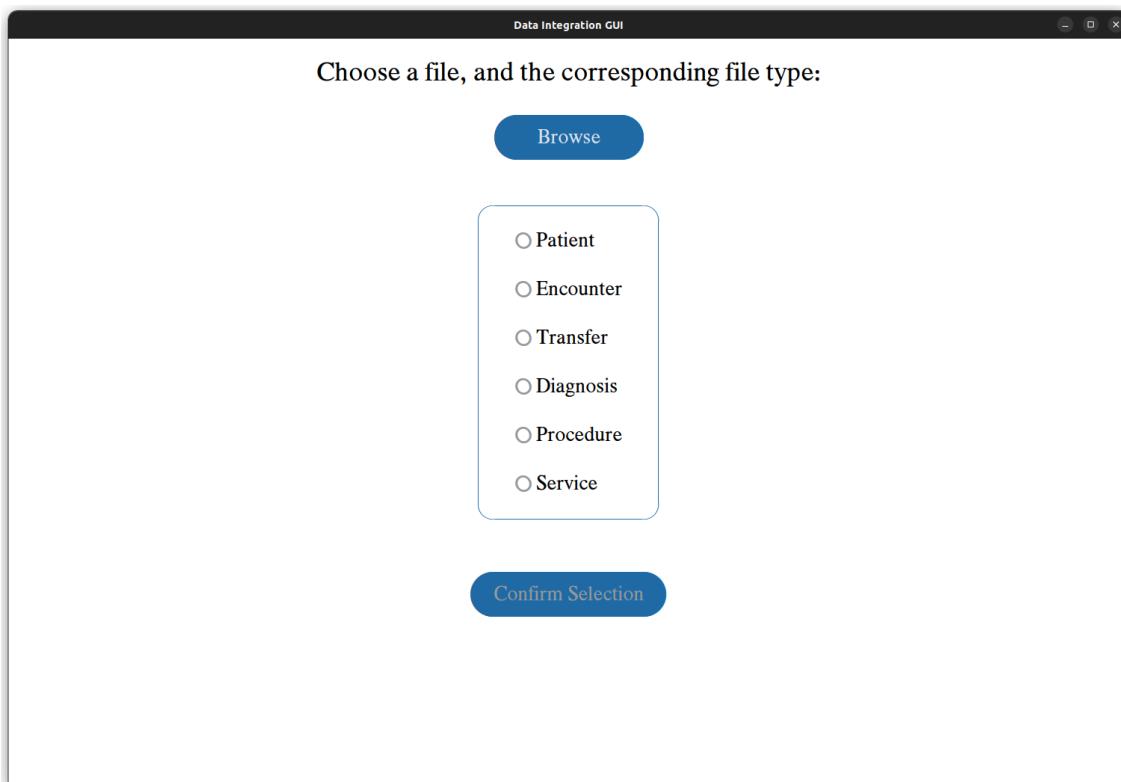


FIGURE 45 – écran de sélection de fichier

Après la sélection du fichier, l'interface lance un processus de mapping automatique, où les colonnes sont détectées selon le type des données, et renommées selon des listes prédéfinies. Dans le cas où certaines colonnes ne sont pas reconnues, le troisième *frame* est emballé. Celui-ci affiche à gauche les noms des colonnes non reconnues dans le fichier importé par l'utilisateur, tandis qu'à droite de chaque nom de colonne se trouve un menu déroulant proposant des suggestions de renommage. Cela offre à l'utilisateur la possibilité de mapper les colonnes du fichier avec des noms plus significatifs ou de les lier à des concepts spécifiques. Le frame contient également un bouton "Reset Mapping" pour réinitialiser le processus de mapping en cas d'erreur, ainsi qu'un bouton "Confirm" pour finaliser le mapping et passer à l'écran suivant.



FIGURE 46 – écran de mapping



FIGURE 47 – Le menu déroulant de l'écran Mapping

Le quatrième et dernier *frame* fait le lien entre l’interface graphique et le conteneur Docker utilisée dans notre projet. Il collecte les informations collectées dans les étapes précédentes et les envoie via une requête HTTP au conteneur Docker, où l’instance NiFi les traitera. De plus, cet écran propose à l’utilisateur de choisir un répertoire sur sa machine locale pour télécharger et sauvegarder au format souhaité les fichiers générés à la fin du traitement par NiFi.

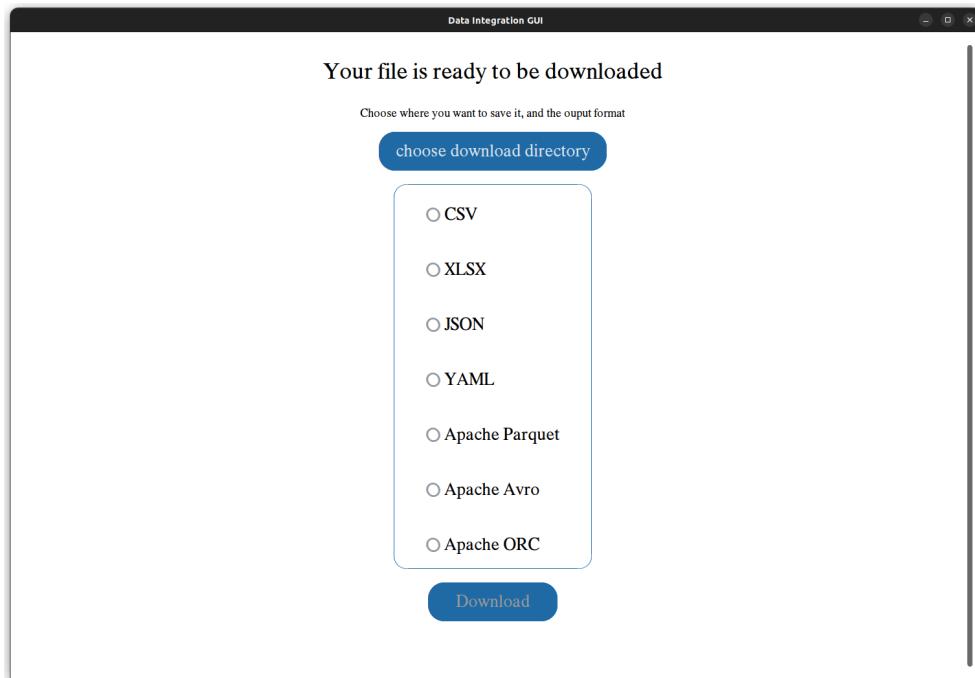


FIGURE 48 – écran de téléchargement

L'ensemble de l'interface graphique a été rigoureusement testé pour assurer sa stabilité, sa performance et sa facilité d'utilisation. Nous avons effectué des tests d'interface utilisateur, identifié et corrigé les éventuels bugs ou erreurs de conception, et nous avons également sollicité les commentaires de notre encadrant et d'un expert du domaine pour améliorer continuellement l'interface.

4.2.3 Explication détaillée de l'interface graphique

4.2.3.1 Le module Hello

Le premier module s'appelle *hello.py*. Il décrit une application GUI développée à l'aide du module **tkinter**. Voici une description générale du code :

1. Tout d'abord, les modules importés :
 - **os** : module pour les opérations système, utilisé pour extraire le nom de fichier.
 - **tkinter** : module principal pour la création d'interfaces graphiques.
 - **customtkinter** : module secondaire contenant des extensions pour tkinter, offrant des améliorations esthétiques.
 - **mappingFrame** : module personnalisé contenant la classe 'MappingFrame'.
 - **fileTypes** : module externe contenant des fonctions pour détecter les types de fichiers.
2. La classe **App** :
 - Hérite de la classe **tk.CTk** (une classe personnalisée basée sur tkinter).
 - Initialise la fenêtre de l'application avec un titre et une taille.
 - Définit des variables membres, dont 'self.file_path' qui stocke le chemin du fichier sélectionné.
 - Appelle les méthodes pour afficher les widgets de sélection de fichier et de type de fichier.
 - Crée des widgets pour le bouton de confirmation et l'étiquette d'avertissement.
3. Méthodes pour afficher les widgets de sélection de fichier :
 - **display_file_selection_widgets()** : crée et affiche l'étiquette de fichier, le bouton de parcours et l'étiquette de chemin.

- **choose_file()** : gère le parcours du fichier, détecte le type de fichier et affiche le chemin sélectionné.
 - **display_file_type_selection_widgets()** : crée et affiche les boutons 'radio' à choix unique pour sélectionner le type de fichier.
4. Méthodes pour gérer la sélection du type de fichier :
 - **radiobutton_event()** : active le bouton de confirmation lorsque le type de fichier est sélectionné.
 - **handle_confirm_button()** : récupère le type de fichier sélectionné et le chemin du fichier, puis crée le cadre du mapping.
 5. Méthodes pour créer et gérer le cadre du mapping :
 - **create_mapping_frame()** : initialise la création du cadre de mapping avec les colonnes du fichier, le chemin du fichier et le type de fichier sélectionné.
 - **forget_file_selection_frame_widgets()** : supprime les widgets du cadre de sélection de fichier pour les remplacer par le cadre de mapping.
 6. Création et exécution de l'application.

Ce premier module définit essentiellement la structure de l'application GUI, y compris les widgets pour la sélection du fichier et du type de fichier, ainsi que la gestion du bouton de confirmation. Il crée également le cadre de mapping, mais il est initialisé en tant que cadre masqué et ne sera affiché que lorsque le bouton de confirmation sera cliqué.

4.2.3.2 Le module MappingFrame

Le module **MappingFrame** est utilisé pour créer un cadre de mapping dans l'interface graphique. Ce cadre permet à l'utilisateur de mapper les noms des colonnes du fichier avec des noms conventionnels.

Dans son constructeur, différentes étapes sont effectuées :

1. La méthode récupère les colonnes de mapping spécifiques en utilisant la fonction **get_mapping_columns_by_type** du module **fileTypes** en fonction du **file_type**, le type de fichier spécifié par l'utilisateur pendant l'étape précédante.
2. Un label d'instruction de mapping est créé et emballé dans le cadre.
3. Si des champs obligatoires sont manquants dans le fichier (vérifié à l'aide de **is_missing_mandatory_field** du module **fileTypes**), un label d'avertissement est créé et aucun mapping n'est effectué.
4. Pour chaque nom de colonne d'origine dans **file_columns**, une instance de la classe **SingleColumnMatchingFrame** est créée et emballée dans le cadre. Ces instances représentent chaque ligne de mapping individuelle.
5. Un cadre de boutons de mapping est créé et emballé dans le cadre inférieur.
6. Une méthode **forget_mapping_frame** est définie pour masquer les éléments du cadre de mapping lorsqu'elle est appelée (au moment où l'étape de mapping est terminée).

Ensuite, le module définit les classes nécessaires pour son fonctionnement. La première classe est appelée **SingleColumnMatchingFrame**. Elle représente une ligne individuelle de mapping d'une colonne.

Dans son constructeur, différentes étapes sont effectuées :

1. Un label est créé pour afficher à gauche le nom de la colonne d'origine.

2. Un menu déroulant ('OptionMenu') est créé pour afficher les options de mapping et l'option "*Leave as is*". Lorsqu'une option est choisie, la méthode **handle_optionmenu_choice()** est appelée pour gérer la logique de mapping.
3. La méthode **handle_optionmenu_choice()** gère les actions lorsqu'une option de mapping est choisie, telles que la mise à jour des dictionnaires de mapping (enlever les options déjà sélectionnées), la désactivation du menu déroulant de la colonne courante (pour éliminer tout risque pour l'utilisateur), l'affichage du nom choisi au milieu, etc.
4. Les méthodes **update_all_option_menus** et **update_own_option_menu** permettent de mettre à jour dynamiquement les options de mapping dans les autres lignes lorsqu'une option est choisie ou mise à jour.

Enfin, le module définit la classe **mappingButtonsBlockFrame**, qui représente un bloc de boutons pour confirmer ou réinitialiser la mapping.

Dans son constructeur, deux boutons sont créés : un bouton de confirmation et un bouton de réinitialisation. Les méthodes **handle_confirm_button()** et **handle_reset_button()** sont définies pour gérer les actions associées à ces boutons, telles que la sauvegarde du dictionnaire de mapping, la réinitialisation du dictionnaire et l'affichage du cadre de téléchargement à la confirmation du mapping.

4.2.3.3 Le module downloadFrame

Le dernier *frame* est une classe appelée **downloadFrame**, qui représente un cadre de téléchargement dans l'interface graphique. Ce cadre permet à l'utilisateur de choisir un répertoire de téléchargement et de télécharger un fichier CSV à partir d'une URL spécifiée.

Le module commence par les importations de modules nécessaires, qu'on a déjà décrits tels que **argparse**, **FileType**, **csv**, **os**, **tkinter**, **customtkinter**, **filedialog**, et **fileTypes**. Et d'autres telles que :

- **requests**, qui permet d'envoyer des requêtes *http/https*, ce qui nous aidera à communiquer avec l'image de NiFi sur Docker.
- **Docker_communication_functions**, qui est un module personnalisé qu'on va détailler un peu plus loin.

Ensuite, le module définit une classe appelée **downloadFrame**. Cette classe représente un cadre de téléchargement pour télécharger un fichier à partir d'un conteneur Docker.

Dans son constructeur, différentes étapes sont effectuées :

1. La méthode initialise des attributs tels que **download_path** (répertoire où le fichier final devrait être enregistré, None pour le moment) et **file_buffer_path** (fichier où les données téléchargées sont stockées en attendant le formatage approprié).
2. Un label d'attente est créé pour afficher un message indiquant que le traitement est en cours dans NiFi.
3. Le fichier source est détecté et converti en un DataFrame CSV en utilisant des fonctions du module **fileTypes**, cette étape est nécessaire car la fonction **POST()** de **requests** (qui dépose des données dans une adresse spécifiée) prend un ensemble limité de formats de data possibles, et CSV en figure.
4. Les données sont envoyées au conteneur Docker à l'aide de fonctions du module **Docker_communication_functions**.
5. Le label d'attente est masqué et les widgets de téléchargement sont affichés à l'aide de la méthode **display_download_widgets**.

6. La méthode **send_data_to_docker_img** est appelée lors de l'initialisation de la classe.

Ensuite, la classe **downloadFrame** définit plusieurs méthodes :

- **send_data_to_docker_img** : Cette méthode envoie les données au conteneur Docker spécifié par l'URL. Elle crée des labels d'attente pour afficher des messages à l'utilisateur. Le fichier est converti en CSV et enregistré dans le répertoire tampon. Ensuite, les données sont envoyées au conteneur Docker à l'aide de la fonction **stream_data_to_docker** du module **Docker_communication_functions**.

```
def send_data_to_docker_img(self , url , source_file_path):
    self.waiting_for_docker_label = tk.CTkLabel(self , font=( "Helvetica" , 40 , "bold" ) , text="Waiting for file to be
        uploaded... ")
    self.waiting_for_docker_label.pack()
    self.waiting_for_docker_label_2 = tk.CTkLabel(self , font=( "Helvetica" , 25) , text="This may take a while" )
    self.waiting_for_docker_label_2.pack()
    #get the file
    file_type = fileTypes.detect_file_type(source_file_path)
    #convert it to csv, and save it in buffer
    df = fileTypes.convert_to_DATAFRAME(source_file_path ,
        file_type)
    fileTypes.convert_to_CSV(df , "./buffer/source_in_xlsx_format")
    #send to docker
    docker_img.stream_data_to_docker(url)#reads from ./buffer
    self.waiting_for_docker_label.pack_forget()
    self.waiting_for_docker_label_2.pack_forget()
    self.display_download_widgets(url)
```

- **display_download_widgets** : Cette méthode affiche les widgets de téléchargement, tels qu'un bouton pour choisir le répertoire de téléchargement, des boutons radio pour sélectionner le type de fichier, et un bouton de téléchargement. Les widgets sont créés à l'aide des classes **CTkButton**, **CTkLabel** et **CTkRadioButton** du module **customtkinter**.
- **handle_choose_directory_button()** : Cette méthode est appelée lorsque le bouton de sélection du répertoire est cliqué. Elle ouvre une boîte de dialogue de sélection du répertoire à l'aide de la fonction **askdirectory** du module **filedialog**. Le répertoire sélectionné est enregistré dans l'attribut 'download_path' et affiché dans un label. Si à la fois le répertoire et le type de fichier ont été sélectionnés, le bouton de confirmation est activé.
- **radiobutton_event** : Cette méthode est appelée lorsqu'un bouton radio est cliqué. Elle vérifie si à la fois le répertoire et le type de fichier ont été sélectionnés. Si c'est le cas, le bouton de téléchargement est activé.
- **Handle_download_button** : Cette méthode est appelée lorsque le bouton de téléchargement est cliqué. Elle utilise la fonction **download_file** du module **Docker_communication_functions** pour télécharger le fichier à partir de l'URL spécifiée et le sauvegarder dans le répertoire de téléchargement sélectionné. Ensuite, le fichier est converti en utilisant la fonction **convert_CSV_to_chosen_type** du module **fileTypes**, en fonction du type de fichier sélectionné.

4.2.3.4 Le module fileTypes

Le module définit plusieurs fonctions liées au types des fichiers d'entrée et de sortie. Voici une description de chaque fonction :

1. **get_mapping_columns_by_type(type)** : Cette fonction prend en paramètre un entier ‘choice’ représentant le type de fichier. Selon la valeur de ‘choice’, la fonction retourne une liste correspondante de colonnes de mapping pour ce type de fichier. De plus, elle enregistre le type de fichier dans le fichier de configuration Nifi. Les types de fichiers disponibles sont :
 - Patient
 - Encounter
 - Transfer
 - Diagnosis
 - Procedure
 - Service
2. **is_missing_elements(list, elements)** : Cette fonction vérifie si des éléments spécifiques sont manquants dans une liste. Elle prend en paramètre une liste et une liste d’éléments à vérifier. Elle retourne *True* si au moins un des éléments est manquant dans la liste, sinon elle retourne *False*.
3. **is_missing_mandatory_field(choice, file_columns)** : Cette fonction vérifie si des champs obligatoires sont manquants dans une liste de colonnes de fichiers donnés. Elle prend en paramètre le type de fichier ‘choice’ et la liste de colonnes **file_columns**. Elle utilise la fonction **is_missing_elements()** pour vérifier la présence des champs obligatoires en fonction du type de fichier. Si des champs obligatoires sont manquants, la fonction retourne *False*, sinon elle retourne *True*.
4. **convert_CSV_to_chosen_type** : Qui permet tout simplement d’appeler la bonne fonction de conversion selon le type choisi. Par exemple :
5. **convert_to_CSV** : Prend un objet dataframe, le convertit en fichier CSV et le sauvegarde dans le répertoire spécifié. Autres fonctions permettent de convertir en XLSX, JSON, YAML, Apache parquet, Apache avro, et Apache ORC.

Ces fonctions sont conçues pour être utilisées dans un contexte où l’on manipule différents types de fichiers et où l’on doit vérifier la présence de certaines colonnes ou champs obligatoires en fonction du type de fichier.

4.2.3.5 Le module Docker_communication_functions :

Ce module inclus deux fonctions supplémentaires : **stream_data_to_docker** et **download_file**. Voici une description de ces fonctions :

- ‘**stream_data_to_docker**’ : Cette fonction envoie des données au conteneur Docker à l'aide de l'URL spécifiée. Elle lit un fichier CSV à partir du chemin “./buffer/source_in_xlsx_format.csv” à l'aide de **pandas**, puis convertit le DataFrame en octets CSV. Ensuite, elle crée un objet *file-like* à un fichier à partir des octets CSV. La fonction envoie ensuite une requête POST (une méthode utilisée dans les protocoles de communication sur Internet pour envoyer des données à un serveur.) à l'URL spécifiée en utilisant les données CSV en streaming. Enfin, elle supprime le fichier “./buffer/source_in_xlsx_format.xlsx” une fois que les données ont été envoyées avec succès.

```

def stream_data_to_docker(url):
    # read from buffer
    csv_file = pd.read_csv("./buffer/source_in_xlsx_format.csv")
    # Convert DataFrame to CSV bytes
    csv_bytes = csv_file.to_csv(index=False).encode('utf-8')
    # Create a file-like object from the CSV bytes
    csv_file_obj = io.BytesIO(csv_bytes)
    # post
    response = requests.post(url, data=csv_file_obj, stream=True)
    # delete buffer
    try:
        os.remove("./buffer/source_in_xlsx_format.xlsx")
        print(f"File ./buffer/source_in_xlsx_format.xlsx deleted
              successfully .")
    except OSError as e:
        print(f"Error deleting file ./buffer/source_in_xlsx_format
              .xlsx: {e}")

```

- ‘download_file’ : Cette fonction télécharge un fichier à partir de l’URL spécifiée et le sauvegarde dans le chemin “file_buffer_path”. Elle envoie une requête GET à l’URL et récupère le contenu de la réponse. Ensuite, elle crée un objet *file-like* à un fichier à partir des données CSV récupérées. La fonction lit ensuite les lignes CSV à l’aide du module csv et les écrit dans un fichier avec le chemin spécifié. Si le téléchargement réussit, un message est affiché. Sinon, un message d’erreur est affiché avec le code d’état de la réponse.

```

def download_file(url, file_buffer_path):
    response = requests.get(url)
    if response.status_code == 200:
        # Get the CSV data from the response content
        csv_data = response.content.decode('utf-8')
        # Create a file-like object from the CSV data
        csv_file = io.BytesIO(csv_data)
        reader = csv.reader(csv_file.splitlines(), delimiter=',')
        with open(file_buffer_path, 'w', newline='') as csvfile:
            writer = csv.writer(csvfile)
            for row in reader:
                writer.writerow(row)
            print(f"CSV file downloaded from {url} and saved to {
                  file_buffer_path}.")
    else:
        print(f"Failed to download CSV file from {url}. Status
              code: {response.status_code}")

```

Ces fonctions font partie du processus de communication avec le conteneur Docker et du téléchargement des fichiers à partir de l’URL spécifiée.

4.2.4 Utilisation de Docker pour la création d'une image NiFi

i Vue détaillée sur Docker

Docker, qui est une plateforme de virtualisation basée sur les conteneurs, a révolutionné la manière dont les applications sont développées, déployées et exécutées. En permettant l'isolation d'applications dans des environnements autonomes et portables, Elle offre une solution efficace pour résoudre les problèmes de compatibilité et de gestion des dépendances.

La principale différence entre les conteneurs Docker et les machines virtuelles traditionnelles réside dans leur approche de virtualisation. Contrairement aux machines virtuelles qui nécessitent l'émulation complète d'un système d'exploitation, chaque conteneur Docker partage le noyau de l'hôte, ce qui permet une exécution plus rapide et une utilisation plus efficace des ressources. Cela signifie que les conteneurs peuvent être démarrés, arrêtés et déplacés très rapidement, offrant ainsi une grande flexibilité et une évolutivité accrue.

Docker repose sur une architecture client-serveur. Le moteur Docker, également appelé "Docker Engine", est le composant central qui gère les conteneurs. Il utilise des images Docker, qui sont des modèles de base pour la création de conteneurs. Une image Docker est un ensemble de fichiers en lecture seule, comprenant un système de fichiers de base et les dépendances nécessaires à l'exécution d'une application. Ces images sont stockées dans des registres Docker, qui peuvent être publics (comme Docker Hub) ou privés, permettant le partage et la distribution des applications et de leurs configurations.

L'utilisation de Docker présente de nombreux avantages. Tout d'abord, Docker facilite la gestion des applications en fournissant un environnement standardisé et reproductible. Les développeurs peuvent créer des conteneurs qui incluent tous les composants nécessaires à l'exécution de leurs applications, éliminant ainsi les problèmes liés aux différences d'environnement entre les machines de développement et de production.

En outre, Docker offre une grande flexibilité en permettant le déploiement d'applications sur divers environnements, qu'il s'agisse de machines physiques, de machines virtuelles ou de services cloud. Les conteneurs Docker sont portables et autonomes, ce qui facilite le déploiement et la migration d'applications entre différents environnements sans avoir à se soucier des dépendances et des configurations spécifiques à chaque plateforme.

Docker facilite également la mise à l'échelle des applications en permettant la réplication des conteneurs. Les conteneurs Docker peuvent être facilement dupliqués pour répondre à une demande accrue, assurant ainsi une haute disponibilité et une performance optimale.



FIGURE 49 – Logo Docker

ii Utilisation de Docker

Dans notre projet, nous avons adopté l'utilisation de Docker pour créer une image NiFi personnalisée, répondant à nos besoins spécifiques. Notre approche reposait sur l'utilisation de deux conteneurs distincts, l'un pour NiFi et l'autre pour Python. Ces deux conteneurs ont été combinés pour former une image Docker complète et fonctionnelle.

L'objectif principal de l'inclusion de Python dans l'image NiFi était de pouvoir exécuter le processeur "ExecuteStreamCommand" qui permet d'exécuter un script Python à chaque itération. Cela nous a permis d'ajouter des fonctionnalités supplémentaires en utilisant la puissance et la flexibilité du langage Python dans notre environnement NiFi.

Pour transférer nos fichiers de script vers l'image Docker, nous avons utilisé la commande "COPY ./source ./target" dans le fichier Dockerfile. Cette étape nous a permis d'inclure nos scripts dans l'image Docker, assurant ainsi leur disponibilité et leur exécution dans l'environnement NiFi. En ayant nos scripts directement inclus dans l'image, nous avons évité les problèmes potentiels liés aux dépendances externes ou aux emplacements de fichiers spécifiques.

En ce qui concerne la configuration de notre environnement Docker, nous avons utilisé Docker Compose. Nous avons spécifié certains paramètres importants dans le fichier de configuration, tels que le numéro de port pour faciliter l'accès à notre application NiFi. Grâce à cette configuration, nous avons pu interagir avec NiFi et accéder à son interface utilisateur via le port défini.

Nous avons également inclus les identifiants de l'utilisateur dans notre configuration Docker Compose. Cela nous a permis de gérer les autorisations et les priviléges d'accès, garantissant ainsi la sécurité et la confidentialité des données manipulées par notre application NiFi. En spécifiant les identifiants de l'utilisateur, nous avons pu restreindre l'accès aux ressources de NiFi uniquement aux utilisateurs autorisés.

Détails techniques :

– Configuration du fichier Docker Compose :

Nous avons commencé par configurer notre fichier Docker Compose (version 3.9) pour définir les services nécessaires au déploiement de NiFi. Nous avons spécifié le service "nifi" en utilisant l'image de base "apache/nifi :latest". Nous avons également défini les ports de communication, en mappant le port 8443 de l'hôte sur le port 8443 du conteneur NiFi, ainsi que le port 5003 de l'hôte sur le port 5003 du conteneur NiFi (on utilise ce numéro de port pour pouvoir recevoir et envoyer des requêtes Http depuis et vers l'interface graphique). De plus, nous avons défini les identifiants de l'utilisateur avec les variables d'environnement "SINGLE_USER_CREDENTIALS_USERNAME" et "SINGLE_USER_CREDENTIALS_PASSWORD". Ces identifiants permettent de gérer l'accès à notre application NiFi de manière sécurisée.

– Construction de l'image NiFi :

Pour construire notre image NiFi personnalisée, nous avons créé un fichier Dockerfile dans le même répertoire que notre fichier Docker Compose. Ce Dockerfile est basé sur l'image "apache/nifi :latest". Nous avons utilisé l'instruction "FROM" pour spécifier cette image de base. Ensuite, nous avons ajouté des instructions supplémentaires pour personnaliser notre image NiFi.

– Construction de l'image NiFi :

Pour construire notre image NiFi personnalisée, nous avons créé un fichier Dockerfile dans

le même répertoire que notre fichier Docker Compose. Ce Dockerfile est basé sur l'image "apache/nifi :latest". Nous avons utilisé l'instruction "FROM" pour spécifier cette image de base. Ensuite, nous avons ajouté des instructions supplémentaires pour personnaliser notre image NiFi.

- Configuration des scripts et des dépendances :

Nous avons copié nos scripts depuis notre répertoire local vers le répertoire cible de l'image NiFi en utilisant l'instruction "COPY". Cela nous a permis d'inclure nos scripts dans l'environnement NiFi et de les rendre accessibles pour l'exécution. Ensuite, nous avons installé les dépendances nécessaires à l'exécution de nos scripts Python en utilisant l'instruction "RUN". Nous avons installé les bibliothèques telles que xlrd, unicodecsv, pandas, datetime, openpyxl, convertdate et numpy en utilisant pip3, le gestionnaire de paquets Python.

- Gestion des volumes :

Un volume est un mécanisme permettant de stocker et de partager des données entre les conteneurs Docker et l'hôte ou entre différents conteneurs.

Nous avons configuré des volumes pour notre conteneur NiFi en utilisant l'instruction "volumes" dans notre fichier Docker Compose. Ces volumes ont été montés pour permettre la persistance des données générées par NiFi dans le répertoire "/opt/nifi/nifi-current/data" du conteneur.

Voici un aperçu de notre fichier DockerFile avec lequel nous avons réussi à build notre image Nifi personnalisée avec Python :

```
FROM apache / nifi : latest
USER root
RUN apt - get update && apt - get install -y python3 python3 - pip
# Install libraries
RUN pip3 install xlrd
RUN pip3 install unicodecsv
RUN pip3 install pandas
RUN pip3 install datetime
RUN pip3 install openpyxl
RUN pip3 install convertdate
RUN pip3 install numpy
```

De même, notre fichier docker compose ressemble à ça :

```
version: "3.9"
services:
  nifi:
    build: ./nifi
    image: apache / nifi : latest
    ports:
      - 8443:8443
      - 5003:5003
    environment:
      - SINGLE_USER_CREDENTIALS_USERNAME=XXXXXXX
      - SINGLE_USER_CREDENTIALS_PASSWORD=YYYYYYY
    volumes:
      - ./nifi_data:/opt/nifi/nifi-current/data:rw
      - /home/bachir/Bureau/S8/HAI823I TER/scripts:/opt/nifi/nifi-current/scripts:rw
```

4.3 Bilan

Tout d'abord, nous avons préparé les données cliniques en utilisant des fichiers structurés et en les soumettant à des prétraitements tels que la validation, l'analyse, le mapping et le nettoyage des données. Pour organiser et moduler les composants de flux de données, nous avons créé deux Process Groups, "Mapping" et "Application des règles", au sein d'Apache NiFi. Ces Process Groups ont permis de structurer le workflow et de faciliter la réutilisation des étapes de mapping et d'application des règles. Ensuite, nous avons chargé les données transformées en les intégrant dans un format standard et facilement exploitable, générant ainsi un fichier du type choisi par l'utilisateur, contenant les informations transformées. Pour assurer la qualité et la validité des données, nous avons également créé un rapport de validation détaillé mettant en évidence les erreurs et les anomalies détectées. Par ailleurs, nous avons développé une interface utilisateur conviviale qui permet à l'utilisateur d'entrer ses fichiers à nettoyer et de récupérer les résultats de manière intuitive. Enfin, nous avons utilisé Docker pour créer une image NiFi personnalisée et combiner deux conteneurs distincts, un pour NiFi et un autre pour Python, offrant ainsi une gestion efficace des conteneurs avec flexibilité, portabilité et possibilité de mise à l'échelle des applications.

4.4 Difficultés rencontrées

Pendant le déroulement du projet, nous avons également été confrontés à certaines difficultés. L'une des principales difficultés était liée à la complexité du processus d'ETL des données cliniques. Comprendre les différentes normes et les défis associés à l'intégration des données a été un défi majeur. De plus, nous avons rencontré des problèmes techniques lors de l'utilisation de l'outil Nifi, notamment dans la configuration des flux de données et la gestion des erreurs.

Un autre défi auquel nous avons été confrontés était la contrainte de temps. En raison de l'échéance serrée du projet, nous avons dû faire face à des pressions pour respecter les délais et finaliser les différentes phases du projet dans les délais impartis. Cela a nécessité une bonne gestion du temps et une planification minutieuse pour surmonter cette difficulté.

4.5 Solutions apportées

Pour surmonter les difficultés mentionnées précédemment, nous avons mis en place plusieurs solutions. Pour faire face à la complexité du processus d'ETL, nous avons investi du temps dans la recherche approfondie et la compréhension des normes et des meilleures pratiques. Nous avons également consulté des experts du domaine et nous avons utilisé des ressources en ligne pour acquérir des connaissances supplémentaires. Cela nous a permis de mieux aborder le processus d'ETL et d'assurer une intégration efficace des données cliniques.

Pour résoudre les problèmes techniques liés à l'utilisation de l'outil Nifi, nous avons fait appel à des ressources en ligne, des forums de discussion et la documentation officielle pour obtenir des réponses et des conseils. Nous avons également échangé des connaissances au sein de l'équipe et nous avons travaillé en étroite collaboration pour résoudre les problèmes rencontrés.

En ce qui concerne la contrainte de temps, nous avons adopté une approche proactive de gestion du projet que nous détaillons ci-après.

5 Gestión du projet

5.1 Organisation

La complexité de notre projet requiert une organisation rigoureuse. C'est pourquoi nous avons dû choisir une méthode de gestion de projets. Nous avons eu une UE nommée "Gestion de projets" encadrée par le professeur "Madalina Croitoru" qui nous a permis de comparer différentes méthodes de gestion de projets.

On a trouvé que la gestion d'un projet faisant appel au processus ETL en utilisant les méthodes en Cascade (en V) serait la plus judicieuse. Ce choix, en raison de la nature relativement stable et bien définie des exigences du projet. La méthode en cascade nous a permis de structurer le processus ETL de manière linéaire, en suivant une séquence d'étapes clairement définies, de l'analyse des besoins à la mise en production. Cela nous a offert une vision globale du projet dès le début et nous a permis de planifier et de coordonner efficacement les différentes phases du processus.

5.1.1 Description de l'organisation

Nous avons établi des réunions hebdomadaires avec l'expert et notre encadrant SERIAI Abdelhak-Djamel, pour fixer les nouveaux objectifs et discuter de l'avancement du projet.

En tant qu'équipe, nous avons maintenu une communication fluide en utilisant des outils de collaboration en ligne. Cela nous a permis de travailler de manière collaborative, de partager des idées rapidement et de coordonner nos actions.

Nous avons identifié les compétences et responsabilités de chaque membre de l'équipe pour une répartition efficace du travail, tout en favorisant la collaboration et l'entraide.

Pour commencer, nous avons établi une planification calendaire du déroulement de notre projet. Nous avons défini des échéances et des objectifs à atteindre tout au long du processus. Ces objectifs hebdomadaires nous ont obligés à adapter en permanence notre approche et à affiner notre travail.

i théorie

Nous nous sommes d'abord penchés sur les concepts théoriques clés pour pouvoir débuter avec des prérequis. Nous avons commencé par approfondir notre compréhension du sujet en réalisant une revue de littérature exhaustive, explorant les différentes facettes du processus d'ETL et les enjeux associés à l'intégration de données cliniques pour pouvoir nous familiariser avec le processus requis pour la réalisation du projet. Parallèlement, nous avons étudié les normes HL7 telles que FHIR, HL7 v2 et v3, qui sont utilisées dans le traitement et la représentation des données cliniques, qui sont les données qui nous intéressent dans ce projet.

ii Pratique

Nous avons par la suite, entamé la partie pratique. Nous avons analysé plusieurs outils d'automatisation du processus ETL pour pouvoir choisir le plus adéquat à notre travail. Après quoi, nous avons entamé les trois phases du processus ETL. Par la suite, nous avons conçu une interface intuitive pour le chargement des données et le téléchargement du fichier résultant par l'utilisateur.

5.1.2 Outils Gestion de Projet

Plusieurs outils ont été utilisés pour faciliter la collaboration, la communication et la gestion de projet, tels que :

- **GitHub** : Un service de gestion de version et de collaboration basé sur Git, qui nous permet de stocker et de partager notre code, de suivre les modifications et de collaborer avec d’autres membres de l’équipe. GitHub nous aide également à organiser notre travail en utilisant des fonctionnalités telles que les issues, les pull requests et les projets.
- **Discord** : Une plateforme de communication en ligne qui facilite la collaboration en temps réel, le partage d’écran et les appels vocaux entre les membres de l’équipe. Discord nous permet d’échanger des idées, de discuter des problèmes rencontrés et de résoudre rapidement les problèmes techniques.
- **Google Docs** : Un service de traitement de texte en ligne qui nous permet de créer, de modifier et de partager des documents texte avec d’autres membres de l’équipe. Google Docs facilite la collaboration en nous permettant de rédiger les idées que notre professeur nous a données et de noter les points importants abordés lors des discussions. Cela nous aide à garder une trace des suggestions et des informations cruciales pour la réussite de notre projet d’intégration de données cliniques.
- **Trello** : Un outil de gestion de projet basé sur le Web qui nous aide à organiser les tâches, à suivre les progrès et à collaborer efficacement avec l’équipe. Trello nous permet de visualiser l’ensemble du projet, de répartir les responsabilités et de suivre l’avancement des différentes étapes de l’intégration des données cliniques.

5.1.3 Outils de recherche

- **ChatGPT** : Un modèle de traitement du langage naturel qui nous a aidés dans la recherche et la documentation concernant notre projet d’intégration de données cliniques. ChatGPT nous a fourni des informations précieuses et des explications détaillées sur les outils tels que NiFi, OpenRefine et Trifecta, ainsi que sur les étapes nécessaires pour utiliser ces outils de manière efficace. Cela a contribué à enrichir notre compréhension des technologies impliquées et à faciliter le processus d’intégration.

5.2 Diagramme de Gantt

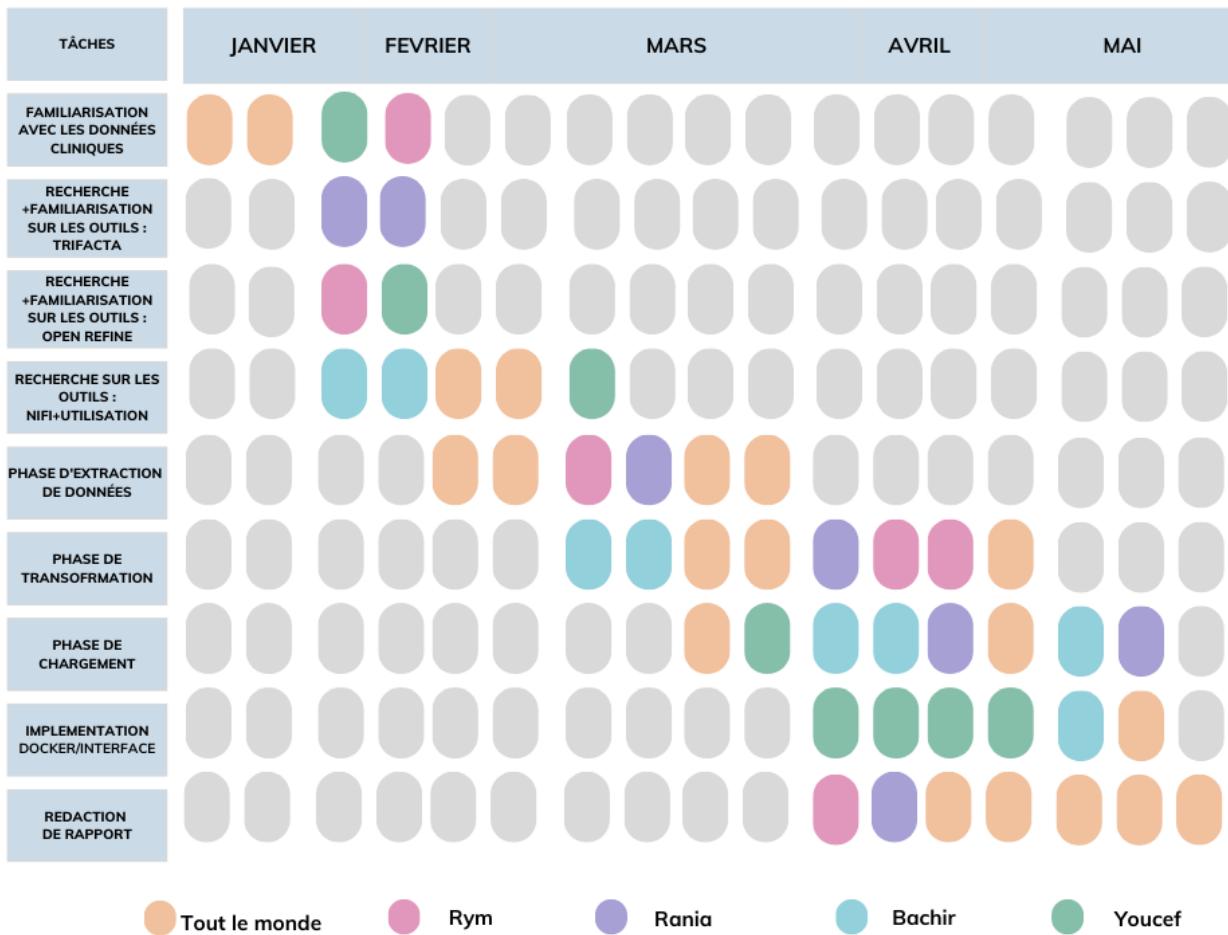


FIGURE 50 – Diagramme de Gantt

6 Conclusion

6.1 Récapitulatif des travaux réalisés

L'objectif principal de ce projet était de fusionner et d'harmoniser des données provenant de différentes sources pour permettre une analyse et une utilisation cohérente dans le domaine médical. Nous avons mis en œuvre un workflow basé sur Apache NiFi pour réaliser cette tâche complexe, après avoir testé d'autres outils pour le faire.

Dans un premier temps, nous avons identifié les différentes sources de données à intégrer, telles que les fichiers contenant des informations sur les patients, les médicaments délivrés, les services de radiologie et les codes de diagnostic, parmi d'autres éléments essentiels.

Ensuite, nous avons procédé à l'étape cruciale de transformation des données. En utilisant les Process Groups d'Apache NiFi, nous avons organisé de manière structurée les différentes étapes du processus d'intégration.

Nous avons également réalisé des prétraitements importants pour assurer la qualité des données. Cela comprenait la validation du domaine d'affaires des fichiers, l'analyse approfondie des structures de données et la définition des règles d'intégration et de mapping nécessaires. L'intégration des données cliniques a nécessité une compréhension approfondie du domaine médical, ainsi qu'une expertise technique dans l'utilisation d'Apache NiFi. Nous avons réussi à résoudre des défis liés à la cohérence des données, aux formats différents et à la complexité des règles d'intégration. Ce projet a été une réussite. Nous avons réussi à fusionner et à harmoniser avec succès des données provenant de différentes sources, ce qui permettra une analyse et une utilisation plus efficaces des données dans le domaine médical. L'utilisation d'Apache NiFi a été essentielle pour gérer le flux de données, appliquer les règles d'intégration et assurer la qualité des données. Il a également mis en évidence l'importance de la collaboration entre les professionnels de la santé et les experts en informatique pour mener à bien des projets d'intégration de données cliniques. La compréhension des besoins spécifiques du domaine médical et l'expertise technique sont des facteurs clés pour garantir le succès de tels projets.

Nous sommes fiers du travail accompli et confiants dans les bénéfices qu'il apportera au domaine médical. Nous espérons que ce rapport servira de référence pour d'autres projets similaires et contribuera à l'avancement de l'intégration.

6.2 Perspectives d'amélioration

Bien que notre travail soit une réussite, nous avons conscience qu'il n'est pas parfait et qu'il pourrait être meilleur. En effet, nous avons pensé à des améliorations qui le rendraient encore plus performant, notamment avec la création d'un espace dédié aux gestionnaires. Nous avons une interface dédiée à l'utilisateur, parmi les perspectives d'amélioration, nous identifions l'implémentation d'un gestionnaire d'authentification permettant de sécuriser l'accès au système. Un tel gestionnaire assurerait une identification et une authentification sécurisées des utilisateurs, garantissant ainsi la confidentialité et l'intégrité des données.

Une autre perspective intéressante est la création et la saisie des règles de manière conviviale. Actuellement, cette tâche nécessite une manipulation directe des données, ce qui peut être complexe pour les utilisateurs non techniques. En développant une interface intuitive, les utilisateurs pourraient créer et saisir facilement des règles en spécifiant les conditions et les actions associées.

La possibilité de modifier et de supprimer des règles existantes constitue également une fonctionnalité clé à ajouter. En offrant cette fonctionnalité, les utilisateurs pourraient ajuster les règles en fonction de l'évolution des besoins ou des contraintes spécifiques de leur domaine d'application.

Une autre perspective d'amélioration concerne la désactivation de règles. Cette fonctionnalité permettrait de suspendre temporairement l'application d'une règle sans la supprimer définitivement, offrant ainsi une plus grande flexibilité dans la gestion des règles.

Nous restons humbles et conscients que chaque projet peut toujours bénéficier d'améliorations continues. Nous sommes ouverts aux suggestions et aux évolutions futures qui pourraient renforcer encore plus notre système, offrant ainsi une valeur ajoutée supplémentaire à nos utilisateurs. Cela témoigne de notre volonté de fournir un système performant, sécurisé et adapté aux besoins de nos utilisateurs, tout en restant ouverts aux innovations futures.

7 Bibliographie

Références

- [1] PowerHealth : <https://powerhealth.com/>
- [2] ETL :
 - <https://fr.wikipedia.org/wiki/Extract-transform-load>
 - <https://actualiteinformatique.fr/data/definition-etl>
- [3] HL7 (Health Level Seven International) :
 - https://fr.wikipedia.org/wiki/Health_Level_7
 - https://en.wikipedia.org/wiki/Health_Level_7
- [4] Fast Healthcare Interoperability Resources (FHIR) :
 - https://fr.wikipedia.org/wiki/Fast_Healthcare_Interoperability_Resources
 - <https://ecqi.healthit.gov/fhir>
 - <https://www.hl7.org/fhir/overview.html>
- [5] Electronic Health Record (EHR) :
 - https://en.wikipedia.org/wiki/Electronic_health_record
 - <https://www.healthit.gov/faq/what-electronic-health-record-ehr>
 - <https://www.hhs.gov/sites/default/files/electronic-health-record-systems.pdf>
 - <https://www.techtarget.com/searchhealthit/definition/electronic-health-record-EHR>
- [6] Apache NiFi :
 - <https://nifi.apache.org/>
 - <https://fr.cloudera.com/products/open-source/apache-hadoop/apache-nifi.html>
 - <https://fr.cloudera.com/about/training/courses/dataflow-flow-managment-with-nifi.html>
- [7] OpenRefine :
 - <https://openrefine.org/>
 - <https://msaby.gitlab.io/tutoriel-openrefine/exploration-et-nettoyage-de-base.html>
- [8] Trifacta :
 - <https://www.trifacta.com/>
 - <https://docs.trifacta.com/display/SS/Home+Page>
- [9] Tkinter : <https://docs.python.org/fr/3/library/tkinter.html>
- [10] Intégration de données cliniques et omiques pour la recherche d'information dans le Dossier-Patient Informatisé :
 - https://www.researchgate.net/publication/280066011_Integration_de_donnees_cliniques_et_omiques_pour_la_recherche_d'information_dans_le_Dossier_Patient_Informatise