

Ministère de l'Enseignement Supérieur et de
la Recherche Scientifique
Université de Manouba
Institut supérieur des Arts Multimédias



MÉMOIRE DE FIN D'ÉTUDES

Préparé en vue de l'obtention du diplôme de License en
Big data et analyse de données

**Mise en place d'une architecture Data Mesh
basée sur la norme HL7 FHIR**

Réalisé par

NADA AMMAR

CHIRAZ BEN BOUBAKER

Encadré par

M : AHMED BEN HAMOUDA - ISAMM

M : HAMED KOUBAA- IT PROGRESS

Année Universitaire : 2022 - 2023

Abstract

This report is written within the context of an internship conducted at IT Progress. Our aim is to implement a Data Mesh architecture based on the HL7 FHIR standard. This approach involves organizing data into autonomous domains, designing domain-specific data products, using data contracts based on HL7 FHIR, and establishing a compatible technical infrastructure. It promotes scalability, data ownership, and interoperability in the healthcare domain, enabling the full utilization of data's value and potential to enhance healthcare.

keywords : Data Mesh architecture, HL7 FHIR, Healthcare data, Autonomous domains, Data products, Graph Database , Datawarehouse, Dashboarding , Recommendation System .

Resumé

Ce rapport est rédigé dans le cadre d'un stage PFE effectué chez IT Progress. Notre objectif est d'implémenter une architecture Data Mesh basée sur la norme HL7 FHIR. Cette approche implique l'organisation des données en domaines autonomes, la conception de produits de données spécifiques à chaque domaine, l'utilisation de contrats de données basés sur HL7 FHIR, et la mise en place d'une infrastructure technique compatible. Cela favorise la scalabilité, la propriété des données et l'interopérabilité dans le domaine de la santé, permettant ainsi de tirer pleinement parti de la valeur des données et de leur potentiel pour améliorer les soins de santé.

Mot clé : Architecture Data Mesh, HL7 FHIR, données de santé, domaines autonomes, produits de données, base de données graphes, entrepôt de données, tableau de bord, système de recommandation.

Dédicaces

Dieu , vous êtes la source de toute vie et de toute lumière. Votre amour inconditionnel et votre compassion ont été une source de réconfort pour moi, m'aidant à surmonter les moments difficiles et à trouver la paix intérieure.

À mes parents, qui ont été mes rochers tout au long de ce parcours éducatif. Votre amour et votre soutien inconditionnels m'ont permis d'espérer toujours encore et encore.

À ma sœur et mon frère, qui ont toujours été là pour moi. Votre présence et votre encouragement m'ont inspiré à donner le meilleur de moi-même.

À mon grand-père décédé, dont j'ai hérité des valeurs, que je les porte en moi comme un héritage précieux. Sa foi m'a appris à être forte, à résister aux épreuves, et à prier chaque matin.

A mes amis ,vos sourires, vos rires et vos épaules ont été un refuge dans les moments difficiles. Vous êtes des âmes précieuses qui ont enrichi ma vie de mille façons.

Finalement à **moi-même** qui m'a appris à être fidèle et à faire confiance, à être drôle et à apprécier la vie, à être positive et à voir le bon dans les autres.

Chiraz Ben Boubaker

Dédicaces

À Dieu, En ce moment de gratitude , je tiens à vous remercier du fond du cœur pour votre soutien inconditionnel. Je vous suis reconnaissante pour m'avoir donné la force de persévérer. Ma réussite est le reflet de votre grâce infinie.

À mes chers parents, Je vous dédie tout mon amour et ma gratitude infinie pour votre soutien inconditionnel tout au long de mon parcours.

Votre amour et vos encouragements sont les piliers de ma réussite. Je vous remercie du fond du cœur pour tout ce que vous faites et continuez de faire pour moi.

À moi-même, Tu es plus forte que tu ne le crois, et tu as en toi la capacité de surmonter tous les défis qui se présentent à toi. Ne cesse jamais de nourrir tes rêves et de poursuivre tes aspirations. Tu mérites le bonheur et la réussite.

À mon cher frère et à ma chère sœur, Que notre relation fraternelle demeure forte et solide, et que nous continuions à nous soutenir mutuellement dans tous les défis qui se présentent à nous. Je suis honorée d'avoir des frère et sœur aussi formidables que vous, et je suis reconnaissante pour chaque instant précieux que nous partageons.

À ma chère grand-mère décédée, Ton héritage de bonté et de force restent gravés dans mon cœur à jamais. Je te porte avec tendresse dans mes souvenirs . Repose en paix, tu me manques énormément.

À mes adorables lapins, Votre présence douce et espiègle illumine mes journées. Merci pour votre amour inconditionnel et votre tendresse sans limite. Que nos moments de jeu et de câlins se multiplient, car vous êtes des êtres spéciaux qui apportent une touche de magie à ma vie.

À mes chers amis , Vous êtes des rayons de soleil dans ma vie. Merci d'être là et de rendre chaque moment inoubliable. Je suis reconnaissante pour notre amitié et je vous chéris profondément.

Nada Ammar

Remerciement

"La gratitude transforme notre passé en une bénédiction, notre présent en une opportunité, et crée une vision pour l'avenir." - Melody Beattie

En exprimant notre gratitude, nous sommes ravis de connaître des profils aussi généreux et qui nous aident à atteindre nos objectifs .

Tout d'abord, nous voulons passer nos larges remerciements à **M.Hamed Koubaa** . Grâce à votre guidance, nous avons pu relever les défis de ce projet ambitieux et de dépasser nos propres limites. Votre confiance en nos compétences nous a motivés à travailler encore plus dur pour atteindre nos objectifs et à nous surpasser dans nos performances.

À notre encadrant pédagogique, **M.Ahmed Ben Hammouda**. Nous tenons à vous remercier chaleureusement pour votre encadrement pédagogique durant notre parcours d'études. Nous sommes reconnaissantes pour votre disponibilité et votre expertise, et Nous sommes fiers de vous avoir eu comme encadrant.

Aux membres du jury,

Nous voudrions vous exprimer notre gratitude pour votre temps et vos efforts consacrés à l'évaluation de notre projet. Vos commentaires et vos critiques constructives nous ont permis de voir notre travail sous un angle nouveau et de l'améliorer.

A nos collègues,

Vous avez été une équipe formidable, toujours prête à relever les défis et à trouver des solutions innovantes. Nous sommes fiers de vous avoir eu comme collègues et Nous sommes reconnaissant pour tout ce que nous avons accompli ensemble.

Encore une fois , nous sommes honorées d'avoir travaillé avec des personnes aussi talentueuses, généreuses et dévouées, et Nous sommes reconnaissantes pour tout ce que nous avons appris grâce à vous. Vos conseils, vos encouragements et votre soutien ont été une source d'inspiration pour nous, et nous sommes fiers de ce que nous avons accompli ensemble.

Table des matières

Introduction générale	1
1 Contexte du projet	3
1.1 Introduction :	3
1.2 Présentation de l'organisme d'accueil :	3
1.2.1 Organisme d'accueil :	3
1.2.2 Domaines d'activités :	4
1.3 Etude de l'existant	4
1.3.1 Présentation de l'existant :	4
1.3.2 Critique de l'existant :	5
1.3.3 Présentation de la solution proposée :	6
1.4 Les Objectifs du projet :	7
1.5 Spécification des besoins :	7
1.5.1 Besoins fonctionnels :	7
1.5.2 Besoins non fonctionnels :	8
1.6 Conclusion :	9
2 Méthodologie de travail	10
2.1 Introduction :	10
2.2 Présentation des Méthodologies :	10
2.2.1 Méthodologie classique :	10
2.2.2 Méthode Agile :	11
2.2.3 Méthode Scrum :	11
2.3 Méthodologie adoptée :	13
2.4 Approche architecturale des données :	13
2.4.1 Entrepôt des données :	13
2.4.2 Data Mesh :	13
2.4.3 Différence entre Entrepôt de données et Data Mesh :	16
2.4.4 Approche Adopté :	17
2.4.5 Conception du Data Mesh :	17
2.5 Choix des technologies et des outils :	18
2.6 Conclusion :	19
3 sprint 1 : Conception et mise en œuvre du domaine individuals	20
3.1 Introduction	20
3.2 Présentation du domaine Individuals	20
3.3 Les outils de gestion de base de données	20
3.4 Outils d' ETL	22
3.5 Architecture globale	22

3.6	Contexte d'une base de donnée orienté graphe	23
3.7	Conception du domaine Individuals	23
3.8	Réalisation	25
3.8.1	Extraction	25
3.8.2	Transformation	27
3.8.3	Load	30
3.9	Conclusion	31
4 sprint 2 : Conception et mise en œuvre du domaine Clinical		32
4.1	Introduction	32
4.2	Les outils de gestion de base des données	32
4.3	Outil d' ETL	33
4.4	Architecture Globale :	33
4.5	Conception du Domaine Clinical :	34
4.6	Modélisation :	35
4.7	Réalisation :	36
4.7.1	Processus ETL pour les dimensions :	36
4.7.2	Génération de la table de fait ClinicalFact :	42
4.7.3	Chargement des données	42
4.8	Conclusion :	44
5 sprint 3 : Conception et mise en œuvre du domaine financial		45
5.1	Introduction :	45
5.2	Outil de gestion de base des données :	45
5.3	Outil d' ETL :	46
5.4	Conception du Domaine Financial :	46
5.4.1	Modélisation :	47
5.5	Mise en place d'un processus ETL :	48
5.5.1	Processus ETL pour les dimensions :	48
5.5.2	Génération de la table de fait FinancialFact :	50
5.5.3	Chargement des données :	51
5.6	Conclusion :	51
6 sprint 4 : Développement d'une application web produite par le département Clinical		52
6.1	Introduction :	52
6.2	Exposition des API :	52
6.2.1	API RESTful :	52
6.2.2	Jhipster :	53
6.2.3	Réalisation :	53
6.2.4	Documentation des APIs :	56
6.2.5	Sécurité des APIs :	56
6.2.6	Performance de l'API :	57
6.3	Les outils de développement :	57
6.3.1	Angular :	58
6.3.2	Spring Boot :	58
6.3.3	conclusion :	58
6.4	Développement des interfaces :	58
6.4.1	Introduction :	58

6.4.2	Objectif de l'application web :	58
6.4.3	Les utilisateurs de l'application web :	59
6.4.4	Les ressources dans l'application Clinical :	60
6.4.5	Tableau de bord :	62
6.5	Conclusion :	62
7	sprint 5 :Assistant de recommandation des médicaments	63
7.1	Introduction :	63
7.1.1	définition :	63
7.1.2	Les types des systèmes de recommandation :	63
7.2	Objectif de l'assistant de recommandation des médicaments :	64
7.3	Ensemble de données utilisé pour développer l'assistant :	65
7.3.1	Préparation des données :	65
7.3.2	Exploration des données :	65
7.4	Comportement des données :	68
7.4.1	Classification :	68
7.4.2	Matrice de factorisation :	69
7.5	Techniques d'apprentissage automatique pour la recommandation des médicaments :	71
7.5.1	Réalisation du premier algorithme :	71
7.5.2	Réalisation du deuxième algorithme :	72
7.5.3	Tester les 2 approches :	72
7.6	Conclusion :	73
Conclusion Générale		74
Table des Annexes		76
Webographie		83
Bibliographie		84

Table des figures

1.1	Logo de société	4
1.2	Une architecture centralisée dans le domaine de santé	5
1.3	Une architecture décentralisée dans le domaine de santé	6
2.4	Les étapes de la gestion de projet avec la méthode Classique	10
2.5	exemple du réalisation d'une Story[1]	11
2.6	Processus Scrum[2]	13
2.7	Architecture Data Mesh [B5]	15
2.8	Comparaison entre les deux architectures	16
2.9	Mise en oeuvre de l'architecture data mesh	17
2.10	logo Google Colab	18
2.11	logo de IntelliJ Idea	18
2.12	logo Google Drive	18
2.13	logo de gitlab	19
2.14	logo Draw.io	19
3.15	Logo de Neo4j	22
3.16	Logo de Python	22
3.17	Architecture Globale Du Domaine Individuals	22
3.18	Conception du domaine Individuals	23
3.19	exemple d'une structure de hl7 fhir	25
3.20	Les bibliothèques python utilisées	26
3.21	Le code d'extraction du noeud Patient	26
3.22	Le code d'extraction du noeud Practitioner	27
3.23	Le code d'extraction du noeud Organization	27
3.24	Le dataset de Patient Après extraction	28
3.25	Le dataset de Patient Après Transformation	28
3.26	Le dataset de Practitioner Après extraction	28
3.27	Le dataset de Practitioner Après Transformation	29
3.28	Le dataset de Organaization Après extraction	29
3.29	Le dataset de Organaization Après Transformation	29
3.30	Portion du Dataset GEOGRAPHIE	29
3.31	Les noeuds dans Neo4j	30
3.32	Exemple d'un nœud Geographie dans Neo4j	30
3.33	Les relations dans Neo4j	31
3.34	la relation LookFor dans Neo4j	31
3.35	un exemple de graphe obtenu dans Neo4j	31
4.36	Logo de postgreSQL	33
4.37	Logo de Python	33
4.38	Architecture Globale du domaine clinical	34
4.39	Conception du modèle Clinical	34

4.40 Partie du code d'extraction du dimension Patient	37
4.41 Partie du code d'extraction du dimension Condition	37
4.42 Partie du code d'extraction du dimension Procedure	37
4.43 Partie du code d'extraction du dimension Medication	38
4.44 Partie du code d'extraction du dimension Encounter	38
4.45 Partie du code d'extraction du dimension Observation	39
4.46 Portion du dataset Patient	39
4.47 Portion du dataset CONDITION	40
4.48 Portion du dataset PROCEDURE	40
4.49 Portion du dataset Immunization	40
4.50 Portion du dataset MEDICATION	40
4.51 Portion du dataset ENOUNTER	41
4.52 Portion du dataset OBSERVATION avant transformation	41
4.53 Portion du dataset OBSERVATION après transformation	41
4.54 Portion du dataset final de factClinical	42
4.55 Partitionnement de factClinical dans PostgreSQL	43
4.56 Temps de réponse avant le partitionnement	44
4.57 Temps de réponse après le partitionnement	44
5.58 Logo de PostgrSQL	45
5.59 logo de Python	46
5.60 Conception du domaine Financial	46
5.61 Partie du code d'extraction du dimension ExplanationOfBenefit	49
5.62 Partie du code d'extraction du dimension Claim	49
5.63 Portion du Dataset EXPLANATION	50
5.64 Portion du Dataset CLAIM	50
5.65 Portion de factFinancial dans PostgreSQL	51
6.66 Popularité de jhipster	53
6.67 Entité Patient dans le back-end	55
6.68 liste des schémas	55
6.69 Exemple de documentation de l'API dans Swagger	56
6.70 Exemple de pagination	57
6.71 Authentification	59
6.72 Inscription	60
6.73 Ensemble des tables existantes	60
6.74 CRUD dans la table Encounter	60
6.75 La recherche dans la table des patients	61
6.76 La table FactClinicals	61
6.77 Détails du champs ProcedureID dans FactClinicals	61
6.78 Vue globale du tableau de bord	62
7.79 Le processus de recommandation des médicaments	64
7.80 les données utilisées pour développer l'assistant de recommandation des médicaments	65
7.81 les 10 conditions de prise de médicaments les plus fréquentes	66
7.82 les 20 médicaments les plus pris	66
7.83 les médicaments recommandés relatifs aux maladies	67
7.84 Un exemple des médicaments recommandés relatif à une maladie	67
7.85 Méthode graphique de Elbow	68
7.86 Visualisation des clusters	69

7.87	Code de la classe MatrixFactorisation	69
7.88	Courbe de perte des données	70
7.89	Résultat de regroupement des médicaments	70
7.90	Présentation d'un exemple de vecteurs	71
7.91	Un exemple de maladie à tester	73
7.92	Exemple de recommandation avec le premier algorithme	73
7.93	Exemple de recommandation avec le deuxième algorithme	73
94	Croissance du téléchargement de la bibliothèque Fhir resources	77
95	Exemple de code utilisant fhir.resources	78
96	Type de jointure en python	78
97	Logo de HL7 fhir	79
98	Les ressources en HL7 fhir[3]	80
99	Logo de Synthéa[4]	81

Liste des tableaux

2.1	Un tableau comparatif entre un data mesh et un Entrepôt de donnée	16
3.2	Déférence entre Neo4j et OrientDB	21
3.3	Conception des noeuds du domaine Individuals	24
3.4	Conception des relations du domaine Individuals	24
4.5	Conception du domaine Clinical	36
5.6	Conception du domaine Financial	48

Liste des acronymes

HL7 : *Health Level 7*

HL7 FHIR : *HL7 Fast Healthcare Interoperability Resource*

API : *Application Programming Interface*

API REST : *Application Programming Interface REpresentational State Transfer*

HTTP : *Hypertext Transfer Protocol*

JSON : *JavaScript Object Notation*

XML : *Extensible Markup Language*

ETL : *Extract-transform-load*

JDL : *JHipster Domain Language*

SGBD : *Système de Gestion de Base de Données*

CRUD : *Create, Read, Update, and Delete*

SQL : *Structured Query Language*

FDW : *Foreign Data Wrapper*

ACID : *Atomicité, Cohérence, Isolation et Durabilité*

GPU : *Graphics Processing Unit*

Introduction générale

Le domaine de la santé est un domaine en évolution constante, où les avancées technologiques ont un impact significatif sur la qualité des soins et la prise de décisions médicales. Les données jouent un rôle important dans ce domaine, en permettant aux professionnels de la santé de mieux comprendre les maladies, les traitements et les résultats, tout en améliorant les résultats pour les patients. Les données médicales peuvent provenir de différentes sources, telles que les dossiers médicaux électroniques, les dispositifs de surveillance, les systèmes de facturation et les systèmes d'information de laboratoire.

Cependant, la collecte et l'analyse de ces données peuvent être un défi majeur en raison de leur diversité, de leur volume important et de leur complexité. Et puis, le traitement des données de santé est crucial pour améliorer les résultats des patients et prendre des décisions éclairées. Les professionnels de la santé ont besoin d'accéder rapidement à des informations précises et fiables pour prendre des décisions en temps réel. Il est donc essentiel de disposer d'une infrastructure de données solide et efficace qui permette d'extraire des informations utiles des données de santé.

Dans ce contexte, nous avons besoin d'une solution qui reflète un changement de mentalité concernant les données et qui met en évidence un aspect clé des données de santé. Le projet de conception et de construction d'un Data Mesh à partir des fichiers HL7 FHIR vise à créer une plateforme de données robuste et interopérable pour améliorer la gestion des données et la prise de décision médicale. En utilisant la norme HL7 FHIR (Fast Healthcare Interoperability Resources), cette plateforme permettra d'intégrer et d'harmoniser les données provenant de différentes sources de données médicales, en offrant une vue globale et unifiée des informations médicales.

Ce rapport, qui se compose de sept chapitres, présentera la synthèse des travaux réalisés dans le cadre de notre projet de fin d'études.

Le premier chapitre de ce rapport est consacré à la présentation du contexte du projet. Dans un premier temps, une présentation détaillée de l'organisme d'accueil sera proposée. Une étude de l'existant sera également réalisée, avec une présentation critique de celui-ci ainsi que la proposition de la solution pour le résoudre. Les objectifs du projet seront clairement énoncés.

Le deuxième chapitre de ce rapport sera consacré à la description des différentes méthodologies ainsi que l'approche architecturale adoptée pour ce projet ainsi que sa conception. Nous détaillerons les choix des technologies et des outils retenus pour le développement de ce projet.

Le troisième chapitre intitulé « Conception et mise en œuvre du domaine Individuals », sera donc essentiel pour comprendre la mise en œuvre technique du domaine Individuals et les choix qui ont été faits pour la gestion de la base de données pour ce département .

Le chapitre 4 de ce document se concentre sur le sprint 2 de la conception et de la mise en œuvre du domaine Clinical. On y trouve une description des outils de gestion de base de données et de l'ETL utilisés pour ce département. La conception du domaine Clinical est détaillée avec une modélisation rigoureuse qui permet une meilleure compréhension de la structure de données. Les étapes de la réalisation sont également expliquées en détail pour donner un aperçu complet du développement du domaine Clinical.

Le cinquième chapitre intitulé « Conception et mise en œuvre du domaine Financial» permettra de comprendre comment le domaine Financial a été conçu et mis en œuvre dans le système de gestion de données. La conception du domaine Financial sera bien détaillé, avec une modélisation des données.

Le chapitre 6 se concentre sur le sprint 4 qui porte sur le développement d'une application web produite par le département Clinical. Dans ce chapitre, on trouve une exposition détaillée des APIs utilisées pour la réalisation du projet. Le développement du tableau de bord en lui-même est expliqué en détail pour donner un aperçu complet de son fonctionnement et de sa conception.

Le septième chapitre intitulé « sprint 5 :Assistant de recommandation des médicaments » étant un produit du domaine «Clinical». Dans ce chapitre, nous trouverons l'objectif de cet assistant produit est ainsi que l'ensemble de données utilisé pour son développement, en passant par la préparation et l'exploration de ces données. Le comportement des données est décrit en détail. Les techniques d'apprentissage automatique pour la recommandation des médicaments sont également abordées avec la réalisation de deux algorithmes différents. Enfin, les deux approches sont testées pour évaluer leurs performances.

Nous clôturerons ce rapport en fournissant une synthèse globale du projet et évoquerons les perspectives d'amélioration envisageables pour l'avenir.

chapitre 1

Contexte du projet

1.1 Introduction :

Une description du cadre du projet est une étape primordiale avant toute conception ou mise en œuvre. Elle permet de fournir une vision claire et détaillée du projet, de faciliter la communication entre toutes les parties prenantes et de garantir la réussite du projet en répondant aux besoins des utilisateurs finaux.

1.2 Présentation de l'organisme d'accueil :

1.2.1 Organisme d'accueil :

ITProgress est une entreprise spécialisée dans les services informatiques, les technologies de pointe, la digitalisation, l'innovation, le domaine de santé, les télécoms et le green tech. Avec plus de 15 ans d'expérience, ITProgress s'est positionnée comme un partenaire de confiance pour accompagner les entreprises dans leur transformation numérique et leur parcours d'innovation. Guidée par une équipe dynamique et passionnée par la technologie, ITProgress met en œuvre des solutions personnalisées pour répondre aux besoins spécifiques de ses clients. L'entreprise se distingue par sa culture d'excellence, de collaboration et d'engagement envers des pratiques durables et responsables, témoignant de son engagement envers l'environnement et la société.



FIGURE 1.1 – Logo de société

1.2.2 Domaines d'activités :

ITProgress opère dans plusieurs domaines d'activités clés. Tout d'abord, l'entreprise fournit des services informatiques complets, allant de la gestion des infrastructures aux solutions logicielles sur mesure. Grâce à son expertise en technologies de pointe, ITProgress propose également des solutions innovantes dans les domaines de la digitalisation, de l'intelligence artificielle et de l'Internet des objets (IoT). De plus, l'entreprise s'est positionnée comme un acteur majeur dans le secteur de la santé en offrant des solutions technologiques avancées pour le secteur médical et le bien-être des patients. Les télécommunications constituent également un domaine d'expertise d'ITProgress, en proposant des solutions de connectivité et de communication adaptées aux besoins des entreprises. Enfin, l'entreprise s'engage activement dans le développement de technologies vertes, en mettant l'accent sur la durabilité et la responsabilité environnementale. Grâce à ces domaines d'activités diversifiés, ITProgress est en mesure de répondre aux défis complexes de ses clients et de contribuer à leur succès durable.

1.3 Etude de l'existant

1.3.1 Présentation de l'existant :

L'architecture dans le domaine de santé est une composante clé des systèmes d'information dans le domaine de la santé. Elle se compose de différents éléments tels que les bases de données, les interfaces et les protocoles de communication. L'architecture centralisée est souvent utilisée dans le domaine de la santé pour stocker et gérer les informations médicales des patients de manière centralisée. Cette approche implique que toutes les données sont stockées dans un seul système de gestion de données.

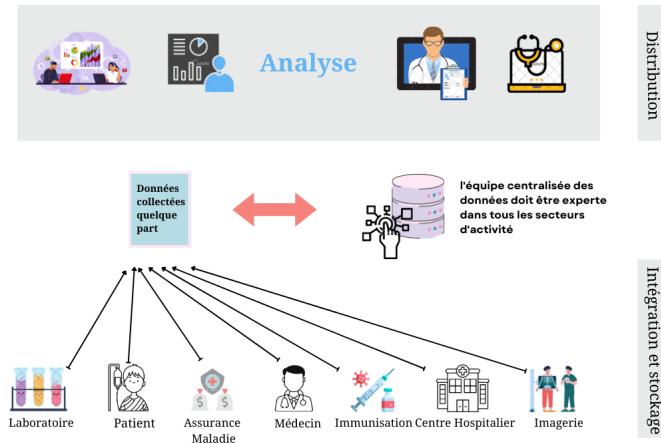


FIGURE 1.2 – Une architecture centralisée dans le domaine de santé

1.3.2 Critique de l'existant :

Les architectures de données centralisées présentent plusieurs problèmes qui peuvent avoir un impact significatif sur la disponibilité, la sécurité et aussi des problèmes liés à la qualité des données présentes dans les modèles analytiques. D'après les statistiques de Gartner [5], d'ici 2025, 80% des organisations cherchant à étendre leur activité numérique échoueront car elles n'adoptent pas une approche moderne de la gouvernance des données et de l'analytique.

Tout d'abord, la dépendance envers un serveur central peut entraîner une perte d'accès aux données en cas de panne du serveur ou de panne technique. De plus, les données stockées sur des serveurs centraux peuvent être vulnérables au piratage ou aux violations, ce qui peut compromettre la sécurité des données stockées. En outre, la création et la maintenance d'une architecture de données centralisée peuvent être coûteuses en raison des coûts de développement, de l'achat de matériel et de la maintenance du serveur central.

En plus, dans le domaine de la santé, les équipes en charge des analyses des données sont souvent distinctes de celles qui sont responsables des systèmes opérationnels. Cependant, cette division des tâches peut entraîner des problèmes tels que des goulots d'étranglement lors de la mise en œuvre des cas d'utilisation et une propriété floue au niveau des sujets/données. En effet, les équipes centralisées en charge des analyses peuvent manquer des connaissances nécessaires du domaine métier des équipes opérationnelles. De plus, les changements apportés au modèle opérationnel peuvent avoir des conséquences inattendues sur le modèle analytique, ce qui peut entraîner des tensions entre les équipes en charge des données et les équipes commerciales. Provenant de différents systèmes, la structure des données de santé n'est pas bien définie, ce qui entraîne une hétérogénéité et une incohérence des informations collectées. Cela rend difficile la comparaison et l'intégration des données.

Enfin, l'architecture doit être en mesure de prendre en compte les problèmes liés à la qualité des données de santé présentes dans les modèles analytiques en une architecture centralisée. Il est donc essentiel de mettre en place une collaboration étroite entre les équipes opérationnelles et analytiques, ainsi qu'une gouvernance rigoureuse des référentiels

et des contrats clairs pour éviter ces problèmes. Selon les recherches de Forrester[6], entre 60 % et 73 % de toutes les données d'une entreprise ne sont pas utilisées pour l'analyse. Dans une enquête récente d'Accenture[7], seules 32 % des entreprises ont déclaré pouvoir réaliser une valeur tangible et mesurable à partir des données, tandis que seulement 27 % ont déclaré que les projets de données et d'analyse produisent des informations et des recommandations hautement exploitables.

1.3.3 Présentation de la solution proposée :

Une solution possible pour résoudre les problèmes liés aux architectures de données centralisées est l'architecture data mesh combinée avec le standard de communication HL7 FHIR (Expliquée dans l'annexe 2). En utilisant l'architecture de données mesh avec HL7 FHIR, les données de santé peuvent être décentralisées et gérées par des domaines autonomes, tout en étant normalisées et interopérables grâce au standard FHIR.

En outre, en combinant l'architecture de données mesh avec HL7 FHIR, les équipes en charge des analyses de données dans le domaine de santé peuvent travailler plus étroitement avec les équipes responsables des systèmes opérationnels pour améliorer la qualité des données et l'efficacité opérationnelle , car chaque domaine est responsable de la gestion de ses propres données et des API permettant l'accès à ces données. Enfin, l'utilisation de HL7 FHIR permet de répondre aux exigences de conformité réglementaire en matière de protection des données de santé, tout en permettant aux utilisateurs de contrôler la façon dont les données sont stockées et utilisées.

En somme, l'architecture Data Mesh associée à HL7 FHIR offre une solution prometteuse pour résoudre les problèmes de centralisation des données et de gouvernance des données dans le domaine de la santé. Elle permet une meilleure collaboration entre les équipes responsables des systèmes opérationnels et des analyses de données, une meilleure réutilisation des données de santé et une réduction des coûts de développement et de maintenance des architectures de données centralisées.

On illustre ce schéma qui présente l'architecture de notre solution :

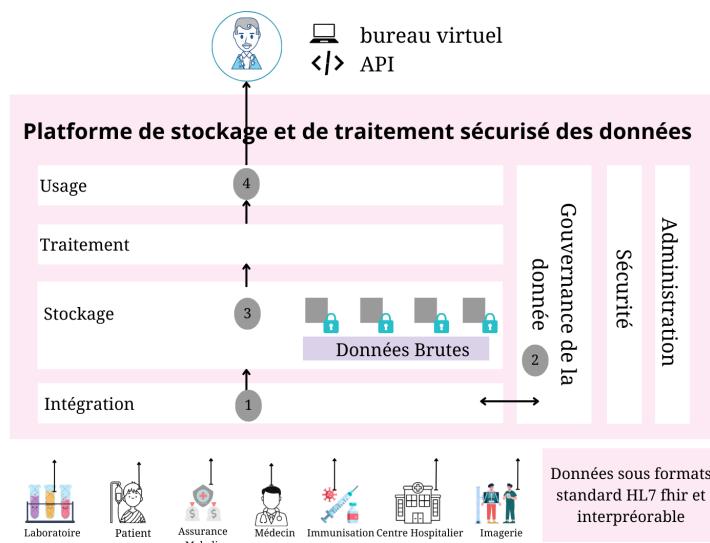


FIGURE 1.3 – Une architecture décentralisée dans le domaine de santé

1.4 Les Objectifs du projet :

L'objectif du projet qui utilise l'architecture Data Mesh et HL7 FHIR dans le domaine healthcare est de permettre une meilleure gestion et utilisation des données. En utilisant l'architecture Data Mesh, les équipes de développement travaillent de manière indépendante et ont une responsabilité claire sur les domaines de données spécifiques. Cela favorise une gouvernance des données plus efficace et une meilleure collaboration entre les équipes techniques et métiers.

En utilisant HL7 FHIR comme standard de données, il est possible d'assurer une interopérabilité entre les différentes applications de santé. Cela facilite la communication entre les différents acteurs du domaine de la santé, y compris les fournisseurs de soins de santé, les patients et les compagnies d'assurance.

En outre, cette solution permet aux équipes de data analystes et data scientistes de manipuler les données plus facilement et efficacement. Les données sont stockées de manière décentralisée, c'est à dire que les équipes peuvent accéder aux données sans dépendre d'une équipe centrale pour accéder à l'ensemble des données.

L'objectif ultime de ce projet est d'améliorer la qualité des soins de santé en permettant une meilleure gestion des données, une meilleure communication entre les différents acteurs du domaine de la santé et une utilisation plus efficace des données pour la prise de décisions.

1.5 Spécification des besoins :

1.5.1 Besoins fonctionnels :

Il est essentiel de spécifier les besoins fonctionnels d'un projet pour assurer sa réussite et sa conformité aux attentes des utilisateurs. Ces besoins décrivent les fonctionnalités et les caractéristiques que le système doit exécuter et les résultats attendus pour chaque action.

- ♣ Collecte et extraction des données : Collecte des données provenant de différents systèmes d'information de santé tels que les dossiers médicaux électroniques, les laboratoires, etc qui doivent être conformes à la norme HL7 FHIR.
- ♣ La mise en place d'une infrastructure de données distribuée : L'architecture Data Mesh repose sur une infrastructure distribuée qui permet de stocker, de traiter et d'analyser les données de manière décentralisée. Il est donc important de mettre en place une infrastructure de données adaptée à cette architecture et qui est composée de différents domaines .
- ♣ Traitement et transformation des données : Les données doivent être nettoyées, normalisées et transformées en utilisant des pipelines de traitement de données (data pipelines) avant d'être stockées dans les référentiels de données spécifiques au domaine. Ces pipelines doivent être évolutifs pour répondre aux besoins spécifiques de chaque équipe.
- ♣ Normalisation et stockage de données : Chaque domaine de l'architecture Data

Mesh doit être capable de normaliser et stocker les données provenant de différents systèmes d'information de santé conformes à la norme HL7 FHIR dans des référentiels de données spécifiques au domaine (domain-specific data warehouses) pour garantir une meilleure qualité des données .

- ♣ Accès et partage des données : Les données doivent être accessibles et partageables entre les équipes de manière sécurisée . Les équipes doivent pouvoir accéder aux données en utilisant des API et en respectant les autorisations d'accès et les politiques de sécurité définies.
- ♣ Analyse et visualisation des données : Les équipes de data analystes et data scientifiques doivent être en mesure d'analyser et visualiser les données en utilisant des outils d'analyse et de visualisation pour obtenir des informations exploitables citant un exemple de produit de données : un tableau de bord interactif qui permet aux professionnels de santé d'accéder aux données de santé facilement et les aider à la prise de décision .
- ♣ Gouvernance et conformité : Les données doivent être gouvernées et gérées conformément aux réglementations de conformité. Les équipes doivent être en mesure de définir et de gérer les politiques de sécurité, les autorisations d'accès et les règles de conformité pour chaque référentiel de données.
- ♣ Maintenance et évolutivité : Le système doit être maintenable et évolutif pour répondre aux besoins changeants des équipes et de l'entreprise. Les équipes doivent être en mesure de déployer rapidement de nouvelles fonctionnalités et de mettre à jour les référentiels de données sans perturber les opérations en cours.

En somme, les besoins fonctionnels pour la mise en place de l'architecture Data Mesh dans le domaine healthcare incluent la mise en place de pipelines de données pour l'extraction, la transformation et le chargement (ETL) de données, la création de services de données basés sur les domaines et la mise en place d'une gouvernance des données décentralisée ainsi que la solution de tableau de bord interactif qui permet aux utilisateurs de visualiser et d'explorer les données .

1.5.2 Besoins non fonctionnels :

Les besoins non fonctionnels sont des exigences qui ne sont pas liées directement aux fonctionnalités du système, mais qui sont nécessaires pour garantir la qualité et les performances du système.

- ♣ Sécurité : Les données médicales sont sensibles et doivent être protégées contre les accès non autorisés. Il est donc essentiel de mettre en place des mesures de sécurité robustes pour protéger les données et empêcher les violations de la confidentialité.
- ♣ Scalabilité : La solution doit être capable de gérer une grande quantité de données et de s'adapter à une utilisation croissante sans sacrifier les performances. Les données doivent être facilement accessibles pour les utilisateurs, même lorsque le volume de données est élevé.
- ♣ Fiabilité : Les données stockées doivent être précises afin que les médecins et les professionnels de la santé puissent prendre des décisions en utilisant ces données. La solution doit être en mesure de garantir que les données soient cohérentes et

mises à jour régulièrement.

- ♣ Interopérabilité : La solution doit être en mesure d'interagir avec différents systèmes d'information en utilisant des normes et des protocoles de l'industrie tels que HL7 FHIR pour permettre l'interopérabilité des données entre différents systèmes et garantir une continuité de soins efficaces.
- ♣ Convivialité : L'interface doit être conviviale et intuitive, de manière à permettre aux utilisateurs de trouver et d'accéder facilement aux données pertinentes pour leur travail.
- ♣ Disponibilité : La solution doit être disponible à tout moment pour les utilisateurs autorisés, en minimisant les temps d'arrêt ou les interruptions de service.
- ♣ Maintenance et évolutivité : La solution doit être facile à maintenir et à mettre à jour, avec des processus d'évolution clairs pour s'adapter aux besoins changeants des utilisateurs et des technologies.

En somme, la prise en compte des besoins non fonctionnels est importante pour s'assurer que le système répond aux exigences de qualité, de performance et de sécurité attendues pour un environnement médical.

1.6 Conclusion :

Dans ce chapitre, nous avons examiné l'existant de notre projet. Nous avons souligné les limites d'architecture centralisée et l'importance de mettre en place une architecture data mesh pour améliorer la qualité et l'accessibilité des données pour les professionnels de la santé. Nous avons également identifié les besoins fonctionnels et non fonctionnels pour la mise en place de l'architecture data mesh et de l'exploitation de données. Ce chapitre permet de mettre en lumière les objectifs et les exigences de ce projet, qui seront développés plus en détail dans les chapitres suivants.

chapitre 2

Méthodologie de travail

2.1 Introduction :

Dans ce chapitre, nous allons discuter de la méthodologie de travail et du concept de Data Mesh utilisés pour la construction de cette architecture conçue pour la prise de décision médicale à partir de fichiers HL7 FHIR. Nous commencerons par expliquer les méthodologies de gestion de projet. Ensuite, nous aborderons le concept de Data Mesh, en expliquant ses principes fondamentaux et comment il peut être appliqué à notre projet

2.2 Présentation des Méthodologies :

2.2.1 Méthodologie classique :

La méthode classique [8] de gestion de projet, également connue sous le nom de cascade, est une approche séquentielle et linéaire de la gestion de projet. Dans cette approche, le projet est divisé en différentes phases, dont chacune est exécutée séquentiellement dans un ordre prédéterminé. Cette figure illustre les phases de réalisation d'un projet avec la méthode Classique :

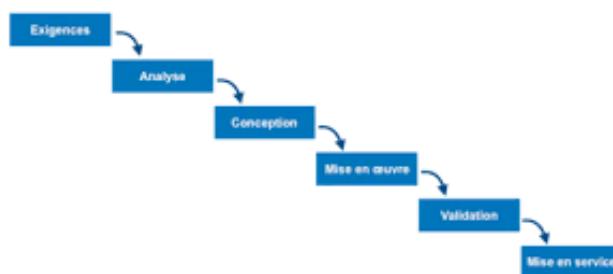


FIGURE 2.4 – Les étapes de la gestion de projet avec la méthode Classique

Dans la méthode classique, la planification du projet est réalisée en amont, et les exigences et les livrables sont définis de manière exhaustive avant le début de chaque phase. Le plan du projet est généralement suivi de manière rigoureuse, et tout changement de direction ou de scope est considéré comme un risque pour le projet. En résumé, la méthode classique de gestion de projet est une approche séquentielle et linéaire, qui met l'accent sur la planification, la documentation et la conformité aux normes. Cette méthode peut être utile pour des projets bien définis et peu susceptibles de changements, mais peut être moins adaptative aux changements de scope et aux exigences évolutives.

2.2.2 Méthode Agile :

La méthode Agile est une approche de gestion de projet itérative et incrémentale, qui se concentre sur la coopération et la communication entre les membres de l'équipe de projet et les parties prenantes. Cette méthode est conçue pour permettre une adaptation continue aux changements, tout en répondant en toute hâte aux besoins des clients. Dans la méthode Agile, le projet est découpé en itérations courtes et itératives, appelées sprints. Chaque sprint est généralement d'une durée de 1 à 4 semaines, et se concentre sur la livraison d'un ensemble de fonctionnalités spécifiques. Les membres de l'équipe de projet travaillent en étroite collaboration pour planifier et exécuter chaque sprint, et effectuent des réunions régulières pour surveiller les progrès et identifier les obstacles potentiels. Cette figure cite l'exemple du réalisation d'une Story.

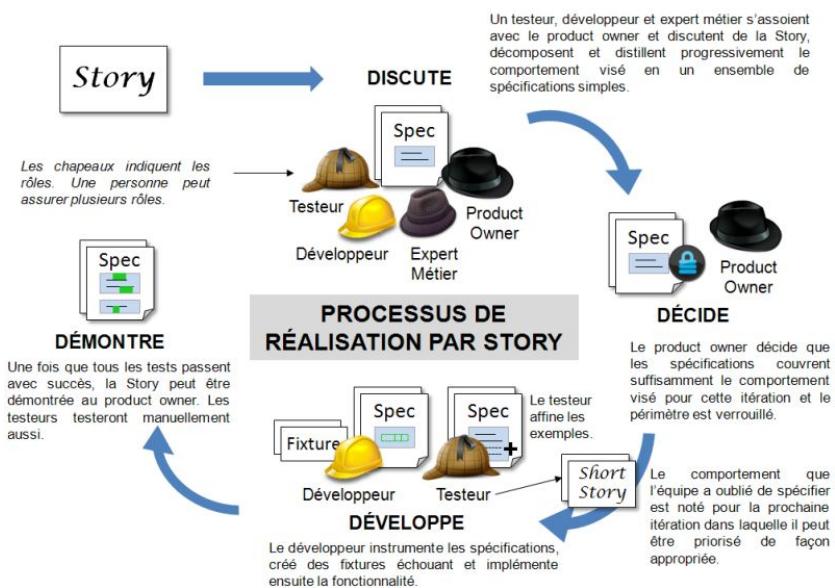


FIGURE 2.5 – exemple du réalisation d'une Story[1]

2.2.3 Méthode Scrum :

Scrum[2] est une méthode agile de gestion de projet qui permet de développer des produits de manière itérative et collaborative. Elle repose sur des cycles de développement

courts appelés "sprints" (généralement de deux à quatre semaines) au cours desquels une partie du produit est développée, testée et livrée. Scrum se base sur des rôles clairement définis (Product Owner, Scrum Master, Équipe de développement), des événements réguliers (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective) et des artefacts (Product Backlog, Sprint Backlog, Increment) pour assurer une collaboration et une transparence optimales entre toutes les parties prenantes du projet.

Les Artefacts de Scrum :

- ♣ **Product Backlog** : c'est une liste ordonnée de toutes les fonctionnalités, tâches ou améliorations à apporter au produit. Les éléments du Product Backlog sont estimés en termes de complexité et de valeur ajoutée, afin de pouvoir prioriser leur réalisation.
- ♣ **Sprint Backlog** : c'est la liste des éléments du Product Backlog que l'équipe de développement s'engage à réaliser pendant le Sprint en cours. Le Sprint Backlog est géré par l'équipe de développement, en collaboration avec le Scrum Master et le Product Owner. Il est mis à jour quotidiennement lors du Daily Scrum.
- ♣ **Increment** : c'est l'ensemble des éléments du Product Backlog qui ont été développés et testés pendant le Sprint en cours, ainsi que toutes les fonctionnalités antérieurement réalisées et qui restent opérationnelles. L'Incrément est l'objectif du Sprint et il doit être potentiellement livrable à la fin de celui-ci. L'objectif est de permettre une livraison fréquente de fonctionnalités à valeur ajoutée pour le client.

Les acteur de Scrum :

Scrum définit trois rôles clés, chacun ayant des responsabilités et des tâches spécifiques :

- ♣ **Product Owner** : c'est la personne qui représente les parties prenantes du projet et qui est responsable de la définition et de la gestion du Product Backlog. Le Product Owner travaille en étroite collaboration avec l'équipe de développement pour s'assurer que le produit réponde aux besoins des utilisateurs et qu'il est livré en temps et en heure.
- ♣ **Scrum Master** : c'est le facilitateur du processus Scrum. Le Scrum Master est responsable de veiller à ce que l'équipe de développement suive la méthode Scrum, en organisant et en animant les différents événements (Daily Scrum, Sprint Planning, Sprint Review, Sprint Rétrospective) et en aidant l'équipe à résoudre les problèmes qui se posent.
- ♣ **Équipe de développement** : c'est le groupe de personnes qui réalise le travail de développement, de test et de livraison du produit. L'équipe de développement est autonome et auto-organisée, ce qui signifie qu'elle est responsable de la planification et de l'exécution de son travail, ainsi que de la gestion du Sprint Backlog. Les membres de l'équipe de développement doivent être pluridisciplinaires et collaborer étroitement pour réaliser les tâches de développement dans le délai imparti.

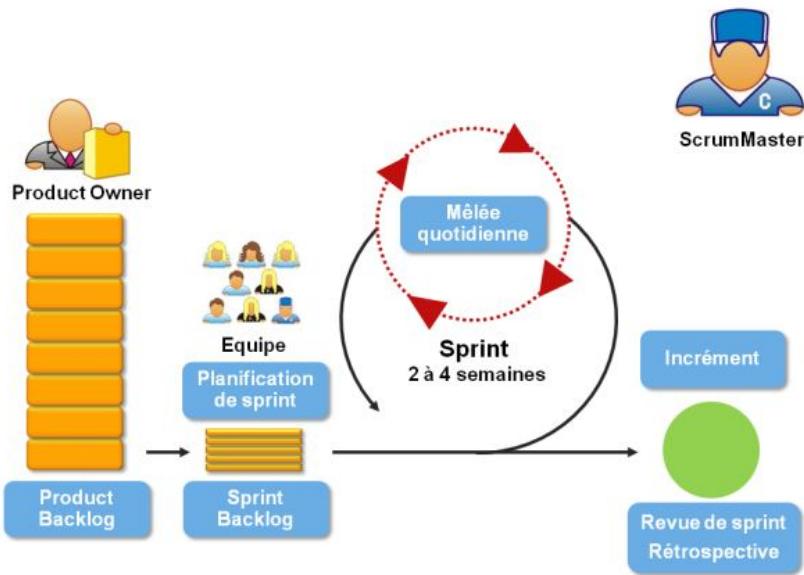


FIGURE 2.6 – Processus Scrum[2]

2.3 Méthodologie adoptée :

Après une recherche approfondie sur les différentes méthodologies de gestion de projet, on a choisi la méthode Scrum pour ce projet. Cette méthode est particulièrement adaptée pour les projets nécessitant une certaine agilité et une forte collaboration entre les membres de l'équipe.

- Product Owner : Hamed Koubaa
- Scrum Master : Hamed Koubaa
- Équipe de développement : Chiraz Ben Boubaker , Nada Ammar

2.4 Approche architecturale des données :

2.4.1 Entrepôt des données :

Un entrepôt de données est un système informatique qui permet de collecter, stocker et gérer de grandes quantités de données provenant de différentes sources, en vue de faciliter leur analyse et leur exploitation pour la prise de décisions. L'entrepôt de données permet d'assembler des données provenant de différentes sources. Ces données sont ensuite traitées, un processus de nettoyage et de standardisation avant tout type de stockage dans l'entrepôt de données, généralement sous forme de tables relationnelles.

2.4.2 Data Mesh :

Le concept de Data Mesh [B1] est une approche émergente de l'architecture de données et de la gestion des données qui a été popularisée par **Zhamak Dehghani**, une

ingénierie de ThoughtWorks, en 2018. La Data Mesh est basée sur le principe selon lequel la gestion des données doit être décentralisée et distribuée aux équipes qui possèdent une connaissance et une expertise approfondies dans un domaine métier spécifique.

Le Data Mesh est un paradigme de décentralisation dans lequel les données sont décentralisées et gérées par des équipes autonomes appelées "**domaines**". Il décentralise la propriété des données, leur transformation en information et leur fourniture. Il vise à accroître l'extraction de valeur des données en éliminant les goulots d'étranglement dans le flux de valeur des données.

Le paradigme Data Mesh est guidé par **quatre principes** qui aident à rendre les opérations de données efficaces à grande échelle. Les mises en œuvre de Data Mesh peuvent différer en termes de portée et du degré d'utilisation de chaque principe.

Principes de Data Mesh : [B2]

- ♣ **La propriété des données doit être décentralisée (Data Ownership)** : Le principe central de l'architecture de données décentralisée orientée domaine est que les données et leurs composants pertinents devraient être possédés, entretenus et développés par les personnes les plus proches de ces données, c'est-à-dire les personnes à l'intérieur du domaine de ces données.
- ♣ **Les données doivent être traitées comme des produits (Data as product)** : Il est important de considérer ce qui donne de la valeur aux données du point de vue de l'organisation, et de penser aux utilisateurs finaux lors de la création de produits de données. Les caractéristiques d'un produit de données incluent une qualité viable, une anticipation des besoins des utilisateurs, une disponibilité sécurisée, un focus sur les objectifs des utilisateurs, une facilité de recherche, et une interopérabilité.
- ♣ **Les équipes doivent partager des normes et des pratiques (Gouvernance fédérée)** : Ce principe vise à fournir un cadre uniifié et une interopérabilité à l'écosystème de produits de données largement indépendants. Son objectif est de faire fonctionner les produits de données autonomes dans un véritable "mesh de données" (et non pas simplement en tant que produits autonomes).
- ♣ **L'architecture doit être conçue pour permettre la découverte et l'utilisation de données (Self service)** : Cela signifie que l'architecture doit être conçue pour permettre aux utilisateurs de découvrir et d'accéder facilement aux données dont ils ont besoin. Les métadonnées doivent être disponibles pour aider les utilisateurs à comprendre les données disponibles et les interdépendances entre les différentes sources de données.

Architecture de Data Mesh [9] :

La figure suivante illustre l'architecture d'un Data Mesh :

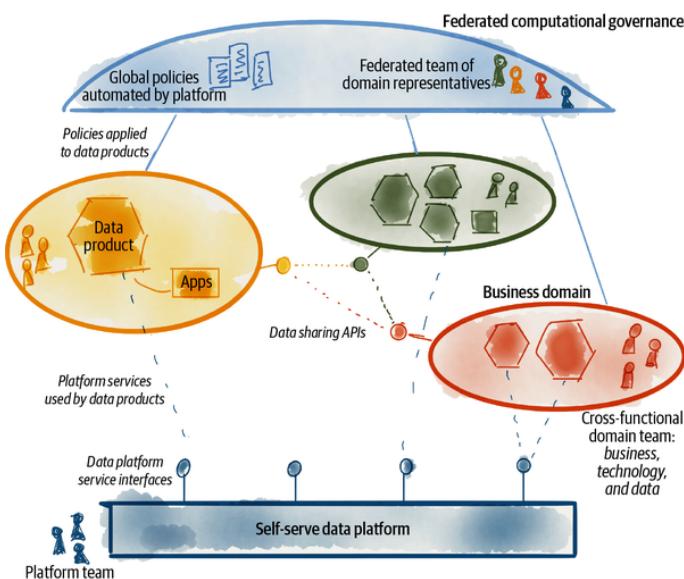


FIGURE 2.7 – Architecture Data Mesh [B5]

Une architecture de maillage de données est une approche décentralisée qui permet aux équipes de domaine de réaliser des analyses de données inter-domaines par elles-mêmes. Au cœur de cette architecture se trouve le domaine avec son équipe responsable et ses données opérationnelles et analytiques. L'équipe du domaine récupère les données opérationnelles et construit des modèles de données analytiques sous forme de produits de données pour effectuer leurs propres analyses. Elle peut également choisir de publier des produits de données avec des contrats de données pour répondre aux besoins en données d'autres domaines. L'équipe du domaine se met d'accord avec d'autres équipes sur les politiques globales, telles que l'interopérabilité, la sécurité et les normes de documentation, au sein d'un groupe de gouvernance fédéré, afin que les équipes du domaine sachent comment découvrir, comprendre et utiliser les produits de données disponibles dans le maillage de données. La plateforme de données autonome et agnostique du domaine, fournie par l'équipe de la plateforme de données, permet aux équipes du domaine de construire facilement leurs propres produits de données et d'effectuer leurs propres analyses de manière efficace. Une équipe facilitatrice guide les équipes du domaine sur la modélisation des données analytiques, l'utilisation de la plateforme de données, et la construction et la maintenance de produits de données interopérables.

Compétences technologiques dans un Data Mesh :

Les compétences technologiques sont généralement axées sur les domaines. Il est important de se responsabiliser socialement pour les données, mais la production d'un produit de données nécessite des capacités technologiques spécifiques. Ces capacités seront déterminées par le domaine, de sorte que le domaine entraîne l'adoption de la capacité technologique. Le domaine devrait décider des technologies de données qui permettent le développement de leur produit de données dans l'environnement de domaine.

2.4.3 Différence entre Entrepôt de données et Data Mesh :

Voici une comparaison sous forme de tableau entre l'entrepôt de données et le Data Mesh :

Critères	Data mesh	Entrepôt de données
Architecture	Architecture distribuée où les données sont stockées et gérées par différentes équipes ou unités d'affaires dans différents emplacements physiques.	Architecture centralisée où toutes les données sont stockées dans un seul emplacement physique, généralement un serveur de base de données.
Gouvernance	Gouvernance décentralisée où chaque équipe est responsable de la qualité de ses données, de leur documentation et de leur accès.	Gouvernance centralisée et contrôlée par une seule équipe responsable de la qualité des données, de leur documentation et de leur accès.
Utilisation	Conçu pour fournir un accès rapide aux données en temps réel pour les applications et les analyses opérationnelles.	Souvent utilisé pour l'analyse en profondeur des données historiques pour les décisions de gestion et les rapports.
Évolutivité	Conçu pour être hautement évolutif, en ajoutant simplement de nouvelles sources de données et en répartissant la charge de travail sur différents emplacements physiques.	Souvent conçu pour une capacité de stockage massive de données, mais peut être limité en termes d'évolutivité horizontale.

TABLE 2.1 – Un tableau comparatif entre un data mesh et un Entrepôt de donnée

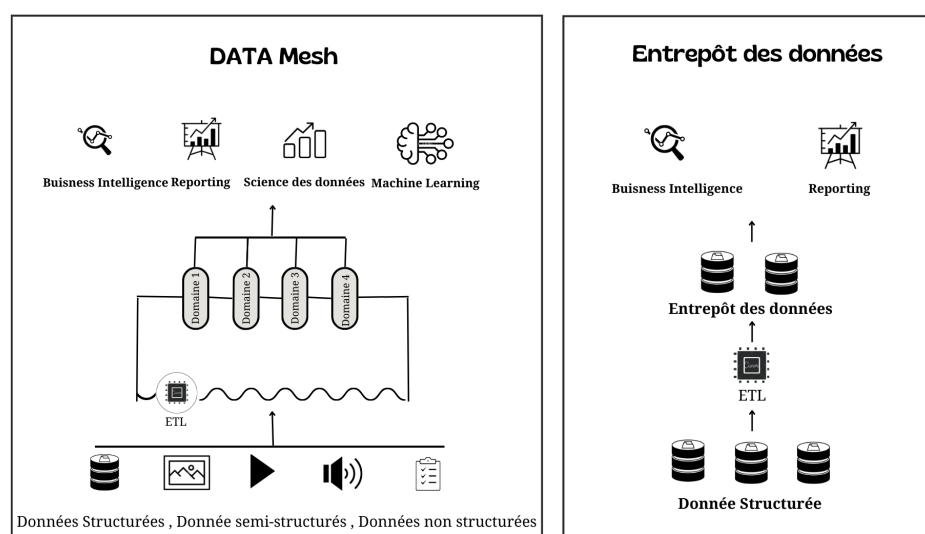


FIGURE 2.8 – Comparaison entre les deux architectures

En résumé, l'entrepôt de données est une architecture centralisée de stockage et de gestion des données gérées par une équipe centrale, tandis qu'un maillage des données est une architecture distribuée où les données sont stockées et gérées par différentes équipes. Le Data Mesh est plus souple et adapté à un environnement distribué et évolutif, tandis que l'entrepôt de données est mieux adapté aux analyses en profondeur des données historiques.

2.4.4 Approche Adopté :

Après une recherche approfondie, on a adopté l'approche du data mesh pour notre gestion de données. Cette approche nous permettra de créer des domaines de données autonomes pour nos différents secteurs, à savoir les domaines "Individuals", "Clinical" et "Financial". Chaque domaine sera dirigé par une équipe propriétaire de produits de données qui seront responsables du développement et de la maintenance des produits de données pour leur domaine spécifique.

2.4.5 Conception du Data Mesh :

Nous allons nous pencher sur la mise en œuvre du Data Mesh dans trois départements différents. Nous allons voir comment la méthodologie et l'approche que nous avons présentées précédemment ont été appliquées dans la pratique.

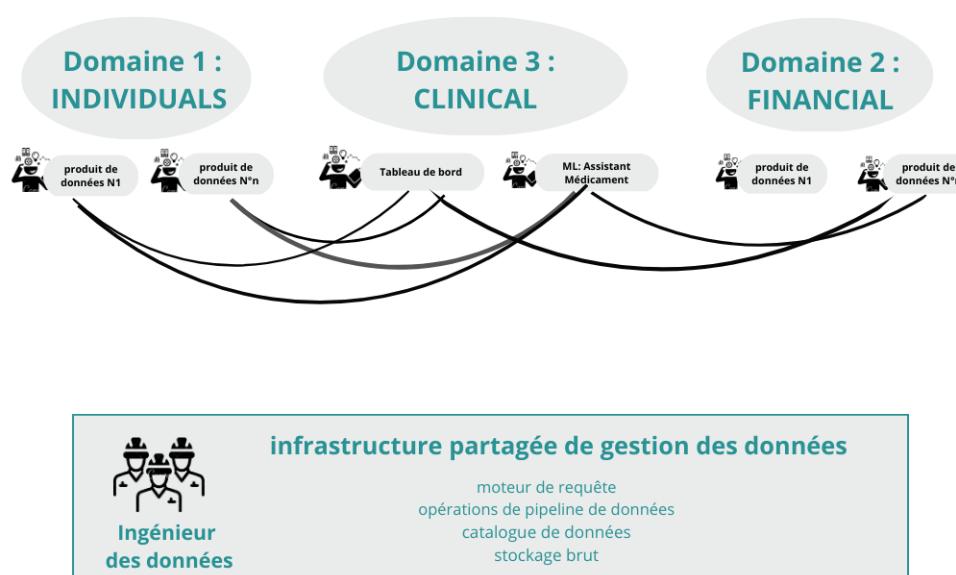


FIGURE 2.9 – Mise en oeuvre de l'architecture data mesh

2.5 Choix des technologies et des outils :

Les outils technologiques sont devenus des éléments incontournables pour la réalisation des projets. Dans le cadre de notre projet, nous avons utilisé différents outils pour communiquer, coder et mener à bien notre projet. Parmi ces outils :

- **Google Colab [10]** : Google Colab est un environnement de développement en ligne basé sur Jupyter Notebook, qui nous permet d'écrire, d'exécuter et de partager du code en temps réel. Nous avons écrit et exécuté notre code en utilisant le langage de programmation Python dans un notebook Colab. Nous avons également utilisé des bibliothèques populaires telles que pandas pour la manipulation des dataframes , pytorch pour entraîner notre modèle , Matplotlib pour visualiser les résultats...



FIGURE 2.10 – logo Google Colab

- **IntelliJ IDEA [11]** : IntelliJ IDEA est un environnement de développement intégré (IDE) pour les langages JVM conçu pour maximiser la productivité des développeurs. Il permet de simplifier les tâches répétitives en offrant des fonctionnalités telles que la complétion intelligente de code, l'analyse statique de code et la refonte de code. Ainsi, les développeurs peuvent se concentrer sur l'aspect créatif du développement de logiciels, rendant l'expérience de développement non seulement productive mais également agréable



FIGURE 2.11 – logo de IntelliJ Idea

- **Google Drive [12]** : Utiliser Google Drive comme environnement de travail nous a permis de stocker, organiser, partager, collaborer et accéder à leurs fichiers et documents de manière efficace et sécurisée, en utilisant des outils de productivité en ligne avancés pour travailler à distance avec nos collègues.



FIGURE 2.12 – logo Google Drive

- **Gitlab** Gitlab est une plateforme de gestion de développement logiciel basée sur Git, offrant un ensemble complet d'outils pour la gestion des dépôts, le suivi des problèmes, l'intégration continue et le déploiement automatique. Il nous permet de travailler de manière collaborative et efficace tout au long du cycle de vie d'un projet.



FIGURE 2.13 – logo de gitlab

- **Draw.io [13]** : Draw.io est un outil de dessin en ligne pour créer des diagrammes professionnels tels que des organigrammes, des diagrammes de flux et des plans d'architecture. L'interface est intuitive et offre des fonctionnalités avancées pour créer des diagrammes complexes. Les fichiers peuvent être stockés localement ou en ligne sur Google Drive, OneDrive, Dropbox et d'autres services, et partagés en toute sécurité avec d'autres.



FIGURE 2.14 – logo Draw.io

2.6 Conclusion :

En conclusion, le choix de la méthodologie et de l'approche pour la gestion des données est une étape primordiale à laquelle tout projet doit passer par. Dans les chapitres suivants, nous explorerons les différentes étapes de mise en œuvre de l'approche de Data Mesh pour les trois domaines identifiés : “Individuals”, “Clinical” et “Financial”. Nous verrons comment les différentes équipes travaillent ensemble pour mettre en œuvre cette approche, les outils et les technologies utilisés pour faciliter la collaboration .

chapitre 3

sprint 1 : Conception et mise en œuvre du domaine individuals

3.1 Introduction

Dans ce deuxième chapitre, nous allons nous pencher sur la mise en œuvre du Data Mesh du domaine Individuals . Nous allons voir comment la méthodologie et l'approche que nous avons présentées dans le chapitre précédent ont été appliquées dans la pratique pour le premier domaine .

3.2 Présentation du domaine Individuals

Le département Individuals vise à mieux comprendre et clarifier les relations entre patients et médecins, et leur répartition géographique. Ces informations peuvent être utilisée pour optimiser ressources et améliorer l'efficacité de l'ensemble du système de santé.

3.3 Les outils de gestion de base de données

♣ Neo4j :

Neo4j[14] est une base de données de graphe open-source qui permet de stocker et de gérer des données avec des relations complexes de manière performante et flexible. Elle utilise un modèle de données de graphe basé sur des nœuds, des relations et des propriétés pour représenter les données. Les requêtes peuvent être effectuées avec le langage de requête Cypher. Neo4j est disponible en plusieurs éditions, dont une gratuite et des éditions payantes avec des fonctionnalités avancées pour les entreprises. Elle est utilisée dans de nombreux domaines, tels que les réseaux sociaux, la recommandation de produits, l'analyse de données, la recherche scientifique et bien d'autres encore. Les données stockées dans Neo4j peuvent être facilement consultées et exploitées en utilisant un langage de requête appelé Cypher, qui permet aux utilisateurs de naviguer dans le graphe et de récupérer des données pertinentes.

♣ OrientDB :

OrientDB[15] est un système de gestion de base de données multi-modèle open-source, créé par OrientDB LTD. Il est conçu pour stocker, manipuler et interroger des données de différents types, y compris les graphes, les documents, les clés/valeurs, les objets géospatiaux et les données en temps réel. OrientDB utilise un langage de requête SQL-like appelé OrientSQL. OrientDB est construit en Java et peut être utilisé à partir de différents langages de programmation, tels que Java, Python et JavaScript. Il prend en charge le clustering, la réPLICATION, la haute disponibilité et la sécurité des données.

♣ Différence entre les deux[16] :

Name	Neo4j	OrientDB
Description	scalable, conforme à ACID, conçue avec une architecture de cluster distribué à haute performance, disponible en versions auto-hébergées et cloud.	Multi-model DBMS (Document, Graph, clé/Valeur) .
Documentation	neo4j.com	orientdb.com
Version initial	2007	2010
Version courante	5.6, March 2023	2.2.22
Licence	Open Source	Open Source
Data scheme	schema-gratuit et schema-optionnel	schema-gratuit
CSV support	Oui	Non
Index secondaire	oui	oui
Langage de quering	Cypher	OrientSQL
Triggers	Oui	Hooks
Méthode de partitionnement	Oui utilisant Neo4j Fabric	Sharding

TABLE 3.2 – Différence entre Neo4j et OrientDB

♣ Base de données adopté :

Le SGBD qu'on va adopter pour répondre aux besoins du département Individuals est Neo4j.

Raison de plus , depuis 2018, le développement d'OrientDB a été suspendu. Le site officiel d'OrientDB est toujours en ligne, mais il n'y a pas eu de mise à jour depuis un certain temps. De plus, la dernière activité sur le dépôt GitHub officiel d'OrientDB remonte à septembre 2021, indiquant que le développement est au ralenti.



FIGURE 3.15 – Logo de Neo4j

3.4 Outils d' ETL

On a décidé d'adopter Python comme outil d' ETL (Extract, Transform, Load) pour le département Individuals dans le cadre de la mise en œuvre du Data Mesh. En utilisant Python pour l'ETL, l'équipe pourra extraire des données des différentes sources, les transformer en fonction des besoins de l'entreprise, puis les charger dans Neo4j pour une analyse plus approfondie.



FIGURE 3.16 – Logo de Python

3.5 Architecture globale

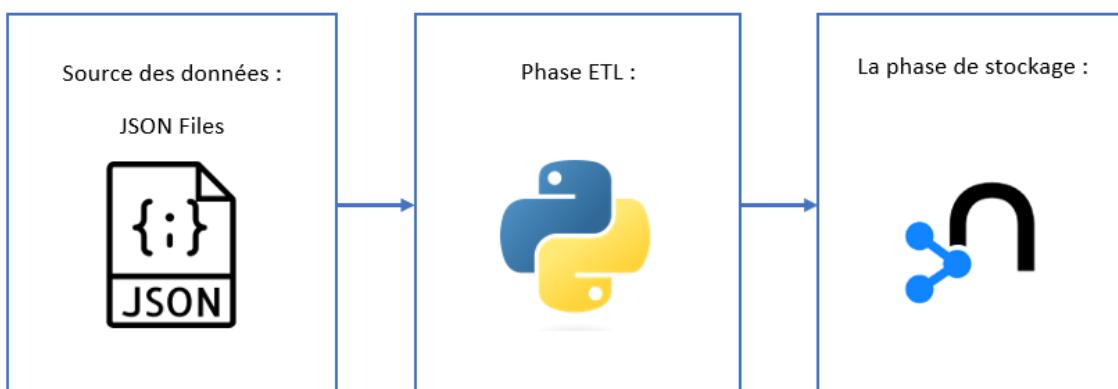


FIGURE 3.17 – Architecture Globale Du Domaine Individuals

3.6 Contexte d'une base de donnée orienté graphe

Une base de données orientée graphe est un type de base de données qui utilise des graphes pour le stockage des données. Les données dedans sont représentées sous forme de noeuds (ou sommets) et de liens (ou arêtes) entre ces noeuds.

Nœuds : les noeuds représentent les entités de données. Chaque noeud peut avoir un ensemble de propriétés, qui sont des valeurs associées à l'entité qu'il représente.

Lien ou relation : Les relations représentent les connexions entre les entités de données, et peuvent avoir des propriétés qui décrivent la nature de la relation.

au sein d'une base de données de graphes, tous les aspects des données (étiquettes, propriétés et types de relation) sont considérés comme des données au sens où ils peuvent être mis à jour en cours d'exécution et il n'est pas nécessaire de redéfinir le schéma de la base de données

3.7 Conception du domaine Individuals

On va essayé après l' ETL de commettre ce graphe :

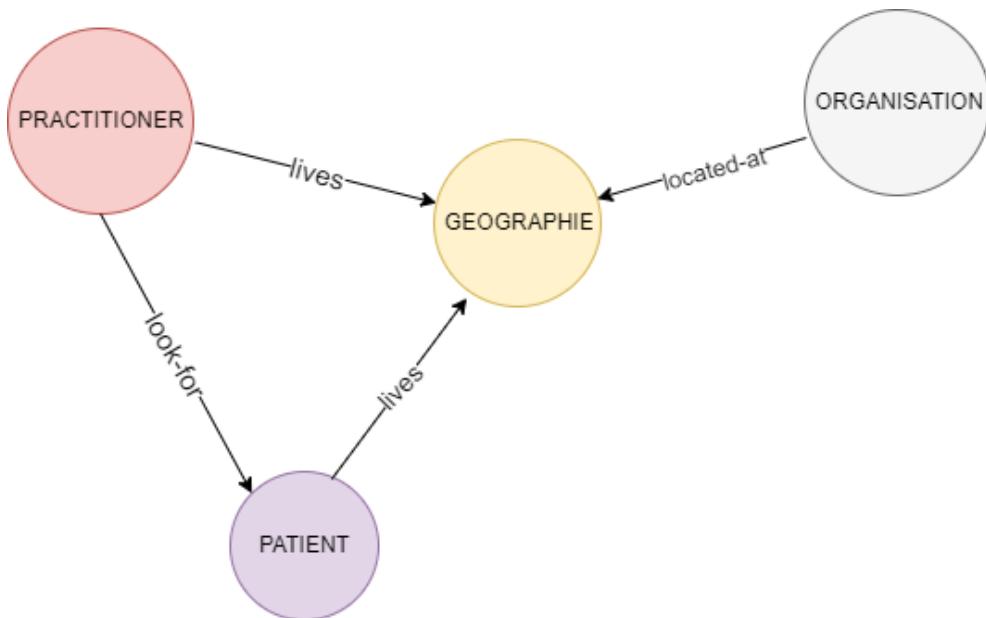


FIGURE 3.18 – Conception du domaine Individuals

Les nœuds seront comme suit :

Nœuds	Définition	Attributs
Patient	Informations démographiques et administratives sur un individu ou un animal recevant des soins ou d'autres services liés à la santé.	PatientUID : varchar(80) PK NameFamily : varchar(30) NameGiven : varchar(30) DoB : Date Gender : varchar(10) Contact : varchar GeographieID : int
Organization	Un regroupement formel ou informel de personnes ou d'organisations reconnu pour le but de réaliser une action collective.	OrganizationUID : varchar(78) Name : varchar(30) Contact : varchar(36) GeographieID : int
Practitioner	Une personne qui est directement ou indirectement impliquée dans la fourniture de soins de santé ou de services connexes.	Practitioner UID : varchar(78) firstname : varchar(30) Name : varchar(30) Gender : varchar(10) Contact : varchar(36) Geoprapgie ID : int
Geographie	Une entité qui regroupe des adresses.	geographieID : int City : varchar(8) State : varchar(10) PostalCode :varchar(8) Country : varchar(4) Line : varchar(30)

TABLE 3.3 – Conception des noeuds du domaine Individuals

On dispose ensuite les relations :

Les relations	Description	Attribut
Located-At	Définir la localisaton du Organization	-
Lives	Définir l'adresse des Patients et des Practitioner	-
Look-for	Définir chaque encounter qu'il a subit un patient	Date

TABLE 3.4 – Conception des relations du domaine Individuals

3.8 Réalisation

3.8.1 Extraction

Le but de cette phase est d'extraire les données qu'on a besoin et qui répond au besoin de notre département.

Problème rencontré

Les fichiers JSON ne sont pas pratiques à utiliser à cause de leur structure complexe, et qu'il y a souvent des données redondantes qui doivent être nettoyées avant de pouvoir les analyser ou les utiliser.

```
{
  "fullUrl": "urn:uuid:0000016d-3a85-4cca-0000-00000001246c",
  "resource": {
    "resourceType": "Practitioner",
    "id": "0000016d-3a85-4cca-0000-00000001246c",
    "identifier": [
      {
        "system": "http://hl7.org/fhir/sid/us-npi",
        "value": "74860"
      }
    ],
    "active": true,
    "name": [
      {
        "family": "Feest103",
        "given": [
          "Delma583"
        ],
        "prefix": [
          "Dr."
        ]
      }
    ],
    "telecom": [
      {
        "system": "email",
        "value": "Delma583.Feast103@example.com",
        "use": "work"
      }
    ],
    "address": [
      {
        "line": [
          "203 MAIN ST"
        ],
        "city": "NORTH READING",
        "state": "MA"
      }
    ]
  }
}
```

FIGURE 3.19 – exemple d'une structure de hl7 fhir

Solution

L'utilisation d'un script Python pour la conversion de fichiers HL7 FHIR de type JSON en dataframes est une étape nécessaire pour la mise en œuvre de la méthodologie Data Mesh dans ce département.

Mise en place de la solution

Cette opération sera réalisée grâce à l'utilisation de la bibliothèque FHIR Ressources (Voir Annexe 1 : Bibliothèque en Python) qui fournit des fonctionnalités pour extraire des données de fichiers FHIR. Les données extraites seront ensuite transformées en dataframes en utilisant des outils tels que Pandas, une bibliothèque Python populaire pour la manipulation de données.

♣ Bibliothèques utilisées :

```

import numpy as np
import pandas as pd
import json
from datetime import date
from tqdm.auto import tqdm
tqdm.pandas()

from fhir.resources.bundle import Bundle
from fhir.resources.patient import Patient
from fhir.resources.practitioner import Practitioner
from fhir.resources.encounter import Encounter
from fhir.resources.organization import Organization

```

FIGURE 3.20 – Les bibliothèques python utilisées

♣ Le script python qu'on a codé permet d'extraire les tables : Practitioner , Patient et Organization :

Ce script lit les fichiers FHIR au format JSON et extrait les informations d'identification et démographiques des patients , Practitioner et Organization. Le script parcourt tous les fichiers dans le répertoire spécifié par fileList et pour chaque fichier, il ouvre le fichier, charge le JSON en objet Python, extrait les ressources du Bundle (ensemble de ressources) et extrait les informations du patient. À la fin de la boucle, les données sont concaténées en une seule DataFrame et un appel de fonction est effectué pour réinitialiser les index.

- Patient** : Voici un extrait du code pour le noeud Patient qui permet d'extraire les attributs à partir des fichiers json de type HL7 FHIR après avoir initialiser le Bundle et les ressources de type Patient :

```

onePatient = Patient.parse_obj(resources[0])

# Patient demographics
onePatientID = onePatient.id
PATIENT.loc[len(PATIENT.index)] = [onePatientID, onePatient.name[0].family, onePatient.name[0].given[0], onePatient.birthDate,
                                     onePatient.gender, onePatient.telecom[0].value, onePatient.address[0].city, onePatient.address[0].country,
                                     onePatient.address[0].state, onePatient.address[0].postalCode, onePatient.address[0].line[0]]

PATIENT = pd.concat([PATIENT, patient], ignore_index = True, axis=0)
PATIENT.reset_index()

```

FIGURE 3.21 – Le code d'extraction du noeud Patient

- Practitioner** : Voici un extrait du code pour le noeud Practitioner qui permet d'extraire les attributs à partir des fichiers json de type HL7 FHIR après avoir initialiser le Bundle et les ressources de type Practitioner :

```

for j in range(len(resPractitioner)):
    oneProc = Practitioner.parse_obj(resPractitioner[j])
    id.append(oneProc.id)
    firstname.append(oneProc.name[0].family)
    name.append(oneProc.name[0].given[0])
    gender.append(oneProc.gender)
    contact.append(oneProc.telecom[0].value)
    city.append(oneProc.address[0].city)
    state.append(oneProc.address[0].state)
    postalCode.append(oneProc.address[0].postalCode)
    country.append(oneProc.address[0].country)

onePractitioner = pd.DataFrame()

onePractitioner['Practitioner UID'] = id
onePractitioner['firstname'] = firstname
onePractitioner['Name'] = name
onePractitioner['Gender'] = gender
onePractitioner['Contact'] = contact
onePractitioner['City'] = city
onePractitioner['State'] = state
onePractitioner['Postal Code'] = postalCode
onePractitioner['country'] = country

PRACTITIONER = pd.concat([PRACTITIONER, onePractitioner], ignore_index = True, axis=0)
PRACTITIONER.reset_index()

```

FIGURE 3.22 – Le code d'extraction du noeud Practitioner

- c. **Organization** : On présente ici une partie du code pour le noeud Organization qui permet d'extraire les attributs à partir des fichiers json de type HL7 FHIR après avoir initialiser le Bundle et les ressources de type **Organization** :

```

id=[]
name=[]
contact=[]
city=[]
state=[]
postalCode=[]
country= []

for j in range(len(resOrganization)):
    oneOrg = Organization.parse_obj(resOrganization[j])
    id.append(oneOrg.id)
    name.append(oneOrg.name)
    if oneOrg.telecom:
        contact.append(oneOrg.telecom[0].value)
    else:
        contact.append('None')
    city.append(oneOrg.address[0].city)
    state.append(oneOrg.address[0].state)
    postalCode.append(oneOrg.address[0].postalCode)
    country.append(oneOrg.address[0].country)

```

FIGURE 3.23 – Le code d'extraction du noeud Organization

3.8.2 Transformation

L'étape de transformation a pour objectif de modifier la structure des tables de données en fonction de nos besoins.

Dans ce cas , on va extraire la table Geographie , et puis on va assigner aux autres tables chacun son ID de géographie . Ensuite , on va achever avec l'étape de standardisation des données et le nettoyage des données .

- a. **Patient** : On présente une partie du Dataset PATIENT avant et après avoir assigner les identifiants de la localisation de chaque Patient et le nettoyage

	PatientUID	NameFamily	NameGiven	DoB	Gender	Contact	City	Country	State	Code Postal	Line
0	5cbc121b-cd71-4428-b8b7-31e53eba8184	Brekke496	Aaron697	1945-12-10	male	555-677-3119	Taunton	US	Massachusetts	02718	894 Brakus Bypass
1	adccf2c3-9dc4-4067-ba23-98982c4875da	Stiedemann542	Aaron697	1946-03-29	male	555-213-2064	Westford	US	Massachusetts	None	378 Krajcik Lodge
2	31191928-6acb-4d73-931c-e601cc3a13fa	Kuvalis369	Abby752	2002-10-24	female	555-598-9552	Wakefield	US	Massachusetts	01880	892 Hoppe Annex
3	67816396-e325-496d-a6ec-c047756b7ce4	Connelly992	Abel832	1999-12-12	male	555-766-3242	Northbridge	US	Massachusetts	None	638 Brakus Union Suite 44
4	b426b062-8273-4b93-a907-de3176c0567d	Heller342	Abraham100	2002-04-15	male	555-803-7962	Somerville	US	Massachusetts	02138	834 Hansen Run

FIGURE 3.24 – Le dataset de Patient Après extraction

Après ↓

	PatientUID	NameFamily	NameGiven	DoB	Gender	Contact	geographield
0	5cbc121b-cd71-4428-b8b7-31e53eba8184	Brekke496	Aaron697	1945-12-10	male	555-677-3119	1
1	adccf2c3-9dc4-4067-ba23-98982c4875da	Stiedemann542	Aaron697	1946-03-29	male	555-213-2064	2
2	31191928-6acb-4d73-931c-e601cc3a13fa	Kuvalis369	Abby752	2002-10-24	female	555-598-9552	3
3	67816396-e325-496d-a6ec-c047756b7ce4	Connelly992	Abel832	1999-12-12	male	555-766-3242	4
4	b426b062-8273-4b93-a907-de3176c0567d	Heller342	Abraham100	2002-04-15	male	555-803-7962	5

FIGURE 3.25 – Le dataset de Patient Après Transformation

- b. **Practitioner** : On présente une partie du Dataset PRACTITIONER avant et après avoir assigner les identifiants de la localisation de chaque individu et le nettoyage

	Practitioner UID	firstname	Name	Gender	Contact	City	State	Postal Code	country
0	0000016d-3a85-4cca-0000-000000000122	Breitenberg711	Bennett146	male	Bennett146.Breitenberg711@example.com	TAUNTON	MA	02780	US
1	0000016d-3a85-4cca-0000-00000000b4d2	Quitzon246	Mason910	male	Mason910.Quitzon246@example.com	RAYNHAM	MA	02767-1973	US
2	0000016d-3a85-4cca-0000-00000000fee2	O'Connell601	Chi716	male	Chi716.O'Connell601@example.com	CHELMSFORD	MA	01824-2712	US
3	0000016d-3a85-4cca-0000-0000000000f0	Heller342	Margene509	female	Margene509.Heller342@example.com	LOWELL	MA	01854	US
4	0000016d-3a85-4cca-0000-0000000016c60	Jakubowski832	Isiah14	male	Isiah14.Jakubowski832@example.com	CHELMSFORD	MA	1824	US

FIGURE 3.26 – Le dataset de Practitioner Après extraction

Après ↓

2995	0000016d-3a85-4cca-0000-00000015b1c	Jaskolski867	Selina302	female	Selina302.Jaskolski867@example.com	2132
2996	0000016d-3a85-4cca-0000-00000007b7a	Tremblay80	Kay203	female	Kay203.Tremblay80@example.com	2133
2997	0000016d-3a85-4cca-0000-00000006540	Parker433	Phillip440	male	Phillip440.Parker433@example.com	2134
2998	0000016d-3a85-4cca-0000-00000009556	Klein929	Shira43	female	Shira43.Klein929@example.com	2135
2999	0000016d-3a85-4cca-0000-000000114fe	Farrell962	Basil991	male	Basil991.Farrell962@example.com	2136

FIGURE 3.27 – Le dataset de Practitioner Après Transformation

- c. **Organization** : On présente une partie du Dataset ORGANIZATION avant et après avoir assigner les identifiants de la localisation de chaque organisation et le nettoyage

	Organization UID	Name	Contact	City	State	Postal Code	country
0	8ad64ecf-c817-3753-bee7-006a8e662e06	MORTON HOSPITAL	5088287000	TAUNTON	MA	02780	US
1	78f24216-8805-3436-ae9f-53aa1cb276b7	PCP116891	None	RAYNHAM	MA	02767-1973	US
2	5f339d06-b1d0-366b-8578-1dffecfe4f07	PCP218528	978-250-9495	CHELMSFORD	MA	01824-2712	US
3	b0e04623-b02c-3f8b-92ea-943fc4db60da	LOWELL GENERAL HOSPITAL	9789376000	LOWELL	MA	01854	US
4	7116c200-6663-3503-8314-1ee16845e5d3	SAINTS WALK-IN MEDICAL CENTER - URGENT CARE AN...	978-458-6868	CHELMSFORD	MA	1824	US

FIGURE 3.28 – Le dataset de Organaization Après extraction

Après ↓

	Organization UID	Name	Contact	geographield
0	8ad64ecf-c817-3753-bee7-006a8e662e06	MORTON HOSPITAL	5088287000	1181
1	78f24216-8805-3436-ae9f-53aa1cb276b7	PCP116891	None	1182
2	5f339d06-b1d0-366b-8578-1dffecfe4f07	PCP218528	978-250-9495	1183
3	b0e04623-b02c-3f8b-92ea-943fc4db60da	LOWELL GENERAL HOSPITAL	9789376000	1184
4	7116c200-6663-3503-8314-1ee16845e5d3	SAINTS WALK-IN MEDICAL CENTER - URGENT CARE AN...	978-458-6868	1185

FIGURE 3.29 – Le dataset de Organaization Après Transformation

- d. **Geographie** : On a générer un nouveau noeud en assemblant toute les adresses qui existent dans les autres noeuds et on a affecté à chacune des un identifiant. On a finalement obtenu une version comme celle-ci :

geographield	City	State	Postal Code	Country	Line
1	Taunton	Massachusetts	2718.0	US	894 Brakus Bypass
2	Westford	Massachusetts	nan	US	378 Krajcik Lodge
3	Wakefield	Massachusetts	1880.0	US	892 Hoppe Annex
4	Northbridge	Massachusetts	nan	US	638 Brakus Union Suite 44
5	Somerville	Massachusetts	2138.0	US	834 Hansen Run

FIGURE 3.30 – Portion du Dataset GEOGRAPHIE

3.8.3 Load

On va charger nos données dans le SGBD Neo4j .

Les nœuds



FIGURE 3.31 – Les noeuds dans Neo4j

Prenons comme exemple un des nœuds Geographie :



FIGURE 3.32 – Exemple d'un nœud Geographie dans Neo4j

Les relations

Notez bien : Afin de créer la relation look-for , on a besoin de l'historique des encouters subit par chaque patient. Etant donnée que nous travaillons dans une architecture Data mesh on peut accéder au données des autres départements . Ainsi , on va importer les données du table Encounter du département Clinical . Cette table sera considérée comme un autre nœud nommé Visit .

Citons l'exemple de la relation Look-for qui relie le nœud practitioner avec le nœud patient ou chaque liaison présente une consultation et elle est attribué à une date bien définis comme suit :



FIGURE 3.33 – Les relations dans Neo4j

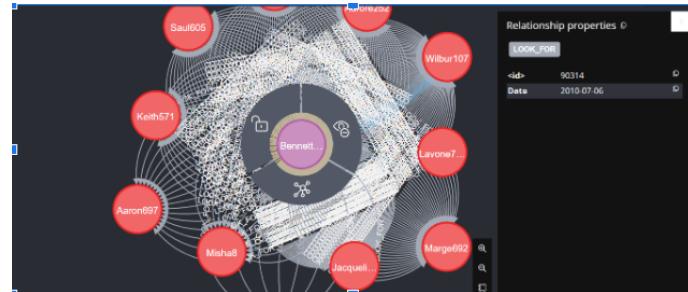


FIGURE 3.34 – la relation LookFor dans Neo4j

Ainsi , on a pu avoir les relations entre les différents noeuds en un seul graphe . Finalement , on a obtenu ce graphe :

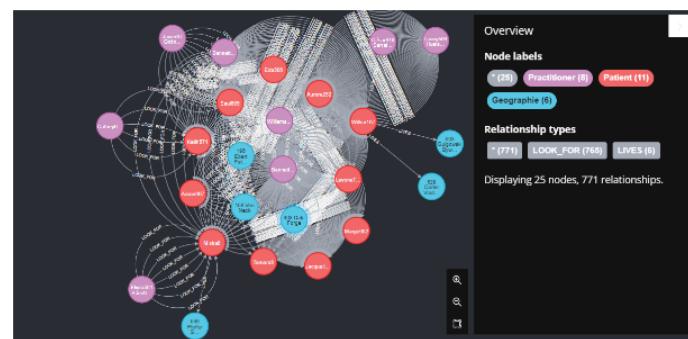


FIGURE 3.35 – un exemple de graphe obtenu dans Neo4j

3.9 Conclusion

En conclusion, le département Individuals nécessite la mise en place d'une base de données orientée graphe pour stocker et gérer efficacement les informations relatives aux patients, tels que leurs données démographiques, leur historique médical ... etc. Cette approche permet de représenter les relations complexes entre les différentes entités de manière plus naturelle et intuitive, et facilite la recherche et l'analyse des données.

chapitre 4

sprint 2 : Conception et mise en œuvre du domaine Clinical

4.1 Introduction

Il s'agit d'un département clé dans le système de gestion de l'information de santé, car il permet de stocker et de gérer les données cliniques du patient, telles que les diagnostics, les traitements, les résultats d'analyses, les procédures médicales, les consultations, les antécédents médicaux, et autres informations cliniques importantes. Ces données cliniques sont utilisées pour prendre des décisions médicales éclairées, assurer la coordination des soins, faciliter la communication entre les professionnels de santé, et garantir une prise en charge globale et de qualité pour les patients.

4.2 Les outils de gestion de base des données

♣ MySql :

est un système de gestion de base de données relationnelle très reconnu possède un Outil graphique Workbench , assure le backup avec l'outil Mysqldump , fournit le sauvegarde grâce à l'outil XtraBackup et son langage de programmation pour les procédures stockées est SQL.

♣ Postgresql [16] :

PostgreSQL est un puissant système de base de données relationnelle objet open source qui utilise et étend le langage SQL et intègre de nombreuses fonctionnalités pour stocker et faire évoluer en toute sécurité les charges de travail de données les plus complexes. Les origines de PostgreSQL remontent à 1986, et il a été activement développé sur la plate-forme principale pendant plus de 35 ans dans le cadre du projet POSTGRES à UC Berkeley. Il a acquis une solide réputation pour son architecture éprouvée, sa fiabilité, son intégrité des données, son ensemble de fonctionnalités robustes, son évolutivité et le dévouement de la communauté open source derrière le logiciel pour fournir constamment des solutions hautes performances et innovantes. PostgreSQL fonctionne sur tous les principaux systèmes d'exploitation, est conforme à ACID depuis 2001 et dispose de puissants modules complémentaires tels que l'extension de base de données géospatiale populaire PostGIS.

♣ Base de donnée adoptée :

Le département Clinical a décidé d'adopter PostgreSQL comme système de gestion de base de données pour la gestion des informations relatives au parcours médical des patients. Avec son support actif de la communauté open-source et sa stabilité, PostgreSQL est un choix judicieux pour la gestion des données cliniques dans le département Clinical. Le choix de PostgreSQL comme base de données pour la gestion du département Clinical peut être justifié par sa structure de données relationnelle, ses performances élevées, sa conformité aux normes de sécurité et de confidentialité, ainsi que son écosystème et son support dans le contexte clinique.



FIGURE 4.36 – Logo de postgresSQL

4.3 Outil d' ETL

On a décidé d'adopter Python comme outil d' ETL (Extract, Transform, Load) pour le département Clinical dans le cadre de la mise en œuvre du Data Mesh. En utilisant Python pour l'ETL, l'équipe pourra extraire des données des différentes sources, les transformer en fonction des besoins de l'entreprise, puis les charger dans Postgresql pour une analyse plus approfondie.



FIGURE 4.37 – Logo de Python

4.4 Architecture Globale :

On vous présente alors l'architecture globale des technologies qui seront utilisé dans ce département

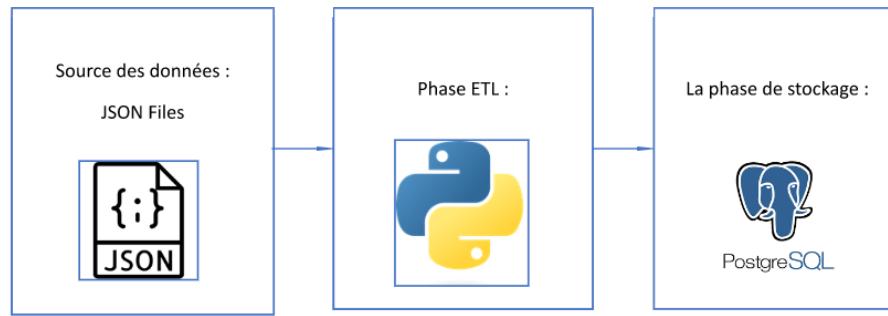


FIGURE 4.38 – Architecture Globale du domaine clinical

4.5 Conception du Domaine Clinical :

Notre objectif est de centraliser et historiser les données de parcours médical des patients dans un entrepôt de données, afin de permettre une analyse approfondie et de faciliter la prise de décisions éclairées. Pour ce faire, nous utilisons une approche basée sur une table de fait sans valeurs mesurées, qui nous permet d'associer différentes dimensions pour représenter les différents aspects des données médicales. Nous allons vous présenter une base solide pour comprendre notre approche de gestion des données médicales. Voici un aperçu :

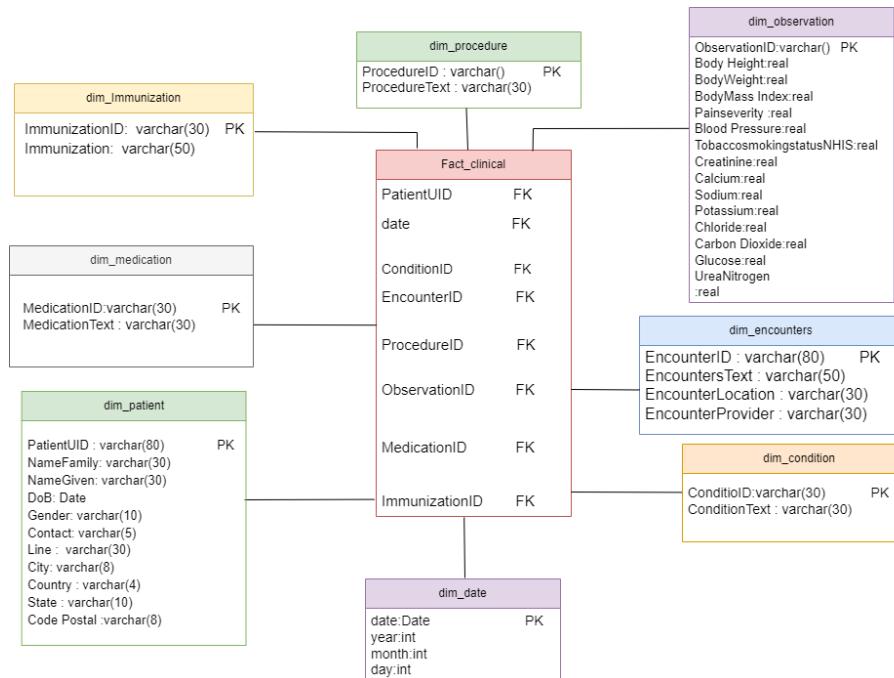


FIGURE 4.39 – Conception du modèle Clinical

4.6 Modélisation :

Nous allons vous présenter les principales tables de dimension qui composent notre entrepôt de donnée, fournissant ainsi une base solide pour comprendre notre approche de gestion des données médicales.

Dimensions	Définition	Attributs
Patient	Informations démographiques et administratives sur un individu recevant des soins ou d'autres services liés à la santé.	patientUID :Character varying nameFamily : Character varying nameGiven :Character varying Birthdate : Date gender : Character varying contact : Character varying line : Character varying city : Character varying country : Character varying city : Character varying state : Character varying postalcode : Character varying
Condition	Une condition clinique, un problème, un diagnostic ou tout autre événement, situation, problème ou concept clinique qui a atteint un niveau de préoccupation.	ConditionID : Serial ConditionText : Character varying
Procedure	Une action qui est ou a été effectuée sur ou pour un patient, un praticien, un dispositif, une organisation ou un lieu.	ProcedureID : Serial ProcedureText : Character varying
Medication	Identifier et définir un médicament.	MedicationID : Serial MedicationText : Character varying
Encounter	Une interaction entre un patient et un ou plusieurs fournisseurs de soins de santé dans le but de fournir des services de soins de santé ou d'évaluer l'état de santé d'un patient.	EncounterID : Serial EncounterText : Character varying EncounterProvider :Character varying EncounterLocation :Character varying
Immunization	Les mesures et les assertions simples faites sur un patient.	ImmunizationID : Serial ImmunizationText : Character varying

Observation	Les mesures et les assertions simples faites sur un patient.	observationID : Serial bodyHeight : float bodyWeight : float bodyMass : float painseverity : float bloodPressure : float tobaccosmokingstatusNHIS : float creatinine : float calcium : float sodium : float potassium : float chloride : float carbonDioxide : float glucose : float ureaNitrogen : float Date : Date Year : int Month : int Day : int
Date	Une dimension qui sert à l'historisation des données .	

TABLE 4.5 – Conception du domaine Clinical

Remarque : Patient.line : Cette composante contient des informations d'adresse telles que le numéro de maison, le numéro d'appartement, le nom de la rue, la direction de la rue, le numéro de boîte postale, des indications de livraison et autres informations similaires.

4.7 Réalisation :

4.7.1 Processus ETL pour les dimensions :

Noté bien : Le problème rencontré qui concerne la complexité de la manipulation des fichiers json de HL7 FHIR a la même solution proposée citée dans le domaine “Individuals”.

Extraction

♣ **Patient :** Voici un extrait du code pour la dimension Patient qui permet d'extraire les données des fichiers json de type HL7 FHIR après avoir distinguer le Bundle et après distigner les ressources de type "Patient" :

```

onePatient = Patient.parse_obj(resources[0])

# Patient demographics
onePatientID = onePatient.id
PATIENT.loc[len(PATIENT.index)] = [onePatientID, onePatient.name[0].family, onePatient.name[0].given[0], onePatient.birthDate,
                                     onePatient.gender, onePatient.telecom[0].value, onePatient.address[0].city, onePatient.address[0].country,
                                     onePatient.address[0].state, onePatient.address[0].postalCode, onePatient.address[0].line[0]]

PATIENT = pd.concat([PATIENT, patient], ignore_index = True, axis=0)
PATIENT.reset_index()

```

FIGURE 4.40 – Partie du code d'extraction du dimension Patient

- ♣ **Condition** : Voici un extrait du code pour la dimension Condition qui permet d'extraire les données des fichiers json de type HL7 FHIR après avoir distinguer le Bundle et les ressources de type "Condition" :

```

conditions = []
conditionOnsetDates = []
for j in range(len(resCondition)):
    oneCondition = Condition.parse_obj(resCondition[j])
    conditions.append(oneCondition.code.text)
    conditionOnsetDates.append(str(oneCondition.onsetDateTime.date()))

onePatConditions = pd.DataFrame()

onePatConditions['ConditionText'] = conditions
onePatConditions['ConditionOnsetDates'] = conditionOnsetDates
onePatConditions['PatientUID'] = onePatientID

CONDITION = pd.concat([CONDITION, onePatConditions], ignore_index = True, axis=0)
CONDITION.reset_index()

```

FIGURE 4.41 – Partie du code d'extraction du dimension Condition

- ♣ **Procedure** : Voici un extrait du code pour la dimension Procedure qui permet d'extraire les données des fichiers json de type HL7 FHIR après avoir distinguer le Bundle et les ressources de type "Procedure" :

```

procs = []
procDates = []
for j in range(len(resProcedures)):
    oneProc = Procedure.parse_obj(resProcedures[j])
    procs.append(oneProc.code.text)
    procDates.append(str(oneProc.performedPeriod.start.date()))

onePatProcs = pd.DataFrame()

onePatProcs['ProcedureText'] = procs
onePatProcs['ProcedureDates'] = procDates
onePatProcs['PatientUID'] = onePatientID

PROCEDURE = pd.concat([PROCEDURE, onePatProcs], ignore_index = True, axis=0)
PROCEDURE.reset_index()

```

FIGURE 4.42 – Partie du code d'extraction du dimension Procedure

- ♣ **Medication** : Voici un extrait du code pour la dimension medication qui permet d'extraire les données des fichiers json de type HL7 FHIR après avoir distinguer le Bundle et les ressources de type "MedicationRequest" :

```

meds = []
medsDates = []
for j in range(len(resMedicationRequest)):
    oneMed = MedicationRequest.parse_obj(resMedicationRequest[j])
    meds.append(oneMed.medicationCodeableConcept.text)
    medsDates.append(str(oneMed.authoredOn.date()))

onePatMeds = pd.DataFrame()

onePatMeds['MedicationText'] = meds
onePatMeds['MedicationDates'] = medsDates
onePatMeds['PatientUID'] = onePatientID

MEDICATION = pd.concat([MEDICATION, onePatMeds], ignore_index = True, axis=0)
MEDICATION.reset_index()

```

FIGURE 4.43 – Partie du code d'extraction du dimension Medication

- ♣ **Encounter** : Voici un extrait du code pour la dimension Encounter qui permet d'extraire les données des fichiers json de type HL7 FHIR après avoir distinguer le Bundle et les ressources de type "Encounter" :

```

encounters = []
encountDates = []
encountLocation = []
encountProvider = []

for j in range(len(resEncounters)):
    oneEncounter = Encounter.parse_obj(resEncounters[j])
    encounters.append(oneEncounter.type[0].text)
    encountLocation.append(oneEncounter.serviceProvider.display)
    if oneEncounter.participant:
        encountProvider.append(oneEncounter.participant[0].individual.display)
    else:
        encountProvider.append('None')
    encountDates.append(str(oneEncounter.period.start.date()))

onePatEncounters = pd.DataFrame()

onePatEncounters['EncountersText'] = encounters
onePatEncounters['EncounterLocation'] = encountLocation
onePatEncounters['EncounterProvider'] = encountProvider
onePatEncounters['EncounterDates'] = encountDates
onePatEncounters['PatientUID'] = onePatientID

ENCOUNTER = pd.concat([ENCOUNTER, onePatEncounters], ignore_index = True, axis=0)
ENCOUNTER.reset_index()

```

FIGURE 4.44 – Partie du code d'extraction du dimension Encounter

- ♣ **Observation** : Voici un extrait du code pour la dimension Observation qui permet d'extraire les données des fichiers json de type HL7 FHIR après avoir distinguer le Bundle et les ressources de type "Observation" :

```

for j in range(len(resObservation)):
    oneObservation = Observation.parse_obj(resObservation[j])
    obsText.append(oneObservation.code.text)
    if oneObservation.valueQuantity:
        obsValue.append(round(oneObservation.valueQuantity.value,2))
        obsUnit.append(oneObservation.valueQuantity.unit)
    else:
        obsValue.append('None')
        obsUnit.append('None')
    obsDate.append(oneObservation.issued.date())

onePatObs = pd.DataFrame()

onePatObs['ObservationText'] = obsText
onePatObs['ObservationValue'] = obsValue
onePatObs['ObservationUnit'] = obsUnit
onePatObs['ObservationDate'] = obsDate
onePatObs['PatientUID'] = onePatientID

OBSERVATION = pd.concat([OBSERVATION, onePatObs], ignore_index = True, axis=0)
OBSERVATION.reset_index()

```

FIGURE 4.45 – Partie du code d'extraction du dimension Observation

Transformation

L'action qu'on va mener plus tard sur les données lors de leur chargement sera l'insertion ou la mise à jour, cette action nécessite forcément la présence d'une clé primaire dans le schéma des données. Donc, après avoir extrait les données pour chaque dimension , nous leur attribuerons une colonne "id" unique pour chaque enregistrement.Ainsi, il est important de compter sur un processus de nettoyage pour garantir que les données soient pertinentes, valides et fiables. Tel que la redéfinition des types et la suppression des doublons et des valeurs nulles .

On a obtenu des datasets comme suit :

♣ **Patient** : On présente ci-dessous une portion du Dataset Patient qu'on a obtenue après transformation :

PatientUID	NameFamily	NameGiven	DoB	Gender	Contact	City	Country	State	Code Postal	Line
5cbc121b-cd71-4428-b8b7-31e53eba8184	Brekke496	Aaron697	1945-12-10	male	555-677-3119	Taunton	US	Massachusetts	2718.0	894 Brakus Bypass
adccf2c3-9dc4-4067-ba23-90982c4875da	Stiedemann542	Aaron697	1946-03-29	male	555-213-2064	Westford	US	Massachusetts	Nan	378 Krajcik Lodge
31191928-6acb-4d73-931c-e01cc3a13fa	Kuvalis369	Abby752	2002-10-24	female	555-598-9552	Wakefield	US	Massachusetts	1880.0	892 Hoppe Annex
67816396-e325-496d-a6ec-c047756b7ce4	Connelly992	Abel832	1999-12-12	male	555-766-3242	Northbridge	US	Massachusetts	Nan	638 Brakus Union Suite 44
b426b062-8273-4b93-a907-de3176c0567d	Heller342	Abraham100	2002-04-15	male	555-803-7962	Somerville	US	Massachusetts	2138.0	834 Hansen Run

FIGURE 4.46 – Portion du dataset Patient

♣ **Condition** : On présente ci-dessous une portion du Dataset CONDITION qu'on a obtenue après transformation :

	Condition ID	ConditionText	Date	PatientUID
0	1	Cardiac Arrest	1965-11-15	5cbc121b-cd71-4428-b8b7-31e53eba8184
1	2	History of cardiac arrest (situation)	1965-11-15	5cbc121b-cd71-4428-b8b7-31e53eba8184
2	3	Body mass index 30+ - obesity (finding)	1977-02-21	5cbc121b-cd71-4428-b8b7-31e53eba8184
3	4	Predabetes	1987-12-21	5cbc121b-cd71-4428-b8b7-31e53eba8184
4	5	Anemia (disorder)	1987-12-21	5cbc121b-cd71-4428-b8b7-31e53eba8184

FIGURE 4.47 – Portion du dataset CONDITION

- ♣ **Procedure** : On présente ci-dessous une portion du Dataset PROCEDURE qu'on a obtenue après transformation :

	Procedure ID	ProcedureText	Date	PatientUID
0	1	Colonoscopy	2010-12-07	5cbc121b-cd71-4428-b8b7-31e53eba8184
1	2	Fecal occult blood test	2010-12-07	5cbc121b-cd71-4428-b8b7-31e53eba8184
2	3	Rectal polypectomy	2010-12-07	5cbc121b-cd71-4428-b8b7-31e53eba8184
3	4	Medication Reconciliation (procedure)	2011-03-07	5cbc121b-cd71-4428-b8b7-31e53eba8184
4	5	Colonoscopy	2012-03-31	5cbc121b-cd71-4428-b8b7-31e53eba8184

FIGURE 4.48 – Portion du dataset PROCEDURE

- ♣ **Immunization** : On présente ci-dessous une portion du Dataset IMMUNIZATION qu'on a obtenue après transformation :

	Immunization ID	Immunization	Date	PatientUID
0	1	Influenza, seasonal, injectable, preservative	2010-03-01	5cbc121b-cd71-4428-b8b7-31e53eba8184
1	2	Influenza, seasonal, injectable, preservative	2011-03-07	5cbc121b-cd71-4428-b8b7-31e53eba8184
2	3	Pneumococcal conjugate PCV 13	2011-03-07	5cbc121b-cd71-4428-b8b7-31e53eba8184
3	4	Influenza, seasonal, injectable, preservative	2012-03-12	5cbc121b-cd71-4428-b8b7-31e53eba8184
4	5	pneumococcal polysaccharide vaccine, 23 valent	2012-03-12	5cbc121b-cd71-4428-b8b7-31e53eba8184

FIGURE 4.49 – Portion du dataset Immunization

- ♣ **Medication** : On présente ci-dessous une portion du Dataset MEDICATION qu'on a obtenue après transformation :

	Medication ID	MedicationText	Date	PatientUID
0	1	Ibuprofen 200 MG Oral Tablet	2016-05-21	5cbc121b-cd71-4428-b8b7-31e53eba8184
1	2	Acetaminophen 325 MG Oral Tablet	2010-07-27	adccf2c3-9dc4-4067-ba23-98982c4875da
2	3	Acetaminophen 325 MG Oral Tablet	2012-01-26	adccf2c3-9dc4-4067-ba23-98982c4875da
3	4	Simvastatin 10 MG	2012-07-20	adccf2c3-9dc4-4067-ba23-98982c4875da
4	5	Simvastatin 10 MG	2013-07-20	adccf2c3-9dc4-4067-ba23-98982c4875da

FIGURE 4.50 – Portion du dataset MEDICATION

♣ **Encounter** : On présente ci-dessous une portion du Dataset ENOUNTER qu'on a obtenue après transformation :

47230	47231	General examination of patient (procedure)	PCP251859	Dr. Basil991 Farrell962	2017-06-22	c588a992-d308-4916-aed5-3894f7e5f6e3
47231	47232	Encounter for check up (procedure)	LAWRENCE GENERAL HOSPITAL	Dr. Cedrick207 Lind531	2018-05-17	c588a992-d308-4916-aed5-3894f7e5f6e3
47232	47233	Encounter for symptom	LAWRENCE GENERAL HOSPITAL	Dr. Cedrick207 Lind531	2018-05-16	c588a992-d308-4916-aed5-3894f7e5f6e3
47233	47234	General examination of patient (procedure)	PCP251859	Dr. Basil991 Farrell962	2018-06-28	c588a992-d308-4916-aed5-3894f7e5f6e3
47234	47235	General examination of patient (procedure)	PCP251859	Dr. Basil991 Farrell962	2019-07-04	c588a992-d308-4916-aed5-3894f7e5f6e3

FIGURE 4.51 – Portion du dataset ENOUNTER

♣ **Observation** : Après l'extraction de la table Observation, on a constaté que les variables mesurées n'étaient pas mises en valeur. Nous allons essayer d'identifier les valeurs mesurées fréquemment utilisées et les transformer en colonnes telles que la masse corporelle...

Cela va être mis en œuvre grâce à un script python qui consiste tout d'abord à sélectionner les valeurs uniques de la colonne "ObservationValue" et puis considérer chacune comme colonne et essayer de la remplir.

Finalement , nous avons abouti à cette transformation :

	ObservationText	ObservationValue	ObservationUnit	ObservationDate	PatientUID
0	Body Height	173.90	cm	2010-03-01	5cbc121b-cd71-4428-b8b7-31e53eba8184
1	Pain severity - 0-10 verbal numeric rating [Sc...]	1.18	{score}	2010-03-01	5cbc121b-cd71-4428-b8b7-31e53eba8184
2	Body Weight	84.17	kg	2010-03-01	5cbc121b-cd71-4428-b8b7-31e53eba8184
3	Body Mass Index	27.83	kg/m2	2010-03-01	5cbc121b-cd71-4428-b8b7-31e53eba8184
4	Blood Pressure	None	None	2010-03-01	5cbc121b-cd71-4428-b8b7-31e53eba8184

FIGURE 4.52 – Portion du dataset OBSERVATION avant transformation

ObservationID	PatientUID	ObservationDate	Body Height	Body Weight	Body Mass Index	Pain severity - 0-10 verbal numeric rating (Score) - Reported	Blood Pressure	Tobacco smoking status NHIS	Creatinine	Calcium	Sodium	Potassium	Chloride	Carbon Dioxide	Glucose	Urea Nitrogen
0	003abc72-e814-4a81-bff1-ce5a54b0e039	2010-01-03	149.19	38.20	17.16	1.29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	003abc72-e814-4a81-bff1-ce5a54b0e039	2010-05-03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	003abc72-e814-4a81-bff1-ce5a54b0e039	2011-01-09	152.95	41.80	17.87	1.67	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	003abc72-e814-4a81-bff1-ce5a54b0e039	2012-01-15	154.38	44.51	18.68	1.38	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	003abc72-e814-4a81-bff1-ce5a54b0e039	2013-01-20	155.02	46.50	19.35	3.61	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
21965	fff2c5e-d4f2-44ae-86aa-9ebc432b0405	2019-01-14	169.62	78.16	27.17	0.75	NaN	NaN	3.37	9.53	143.41	4.03	107.98	24.56	105.30	9.72
21966	fff2c5e-d4f2-44ae-86aa-9ebc432b0405	2019-03-11	169.62	78.16	27.17	0.02	NaN	NaN	4.68	10.07	141.43	4.04	105.91	20.06	116.95	15.93
21967	fff2c5e-d4f2-44ae-86aa-9ebc432b0405	2019-04-08	169.62	78.16	27.17	1.78	NaN	NaN	3.69	9.65	136.08	5.06	107.94	28.75	121.46	18.29
21968	fff2c5e-d4f2-44ae-86aa-9ebc432b0405	2019-08-05	169.62	78.16	27.17	2.83	NaN	NaN	4.36	9.16	141.78	3.82	104.64	24.70	100.18	13.31
21969	fff2c5e-d4f2-44ae-86aa-9ebc432b0405	2019-09-02	169.62	78.16	27.17	0.50	NaN	NaN	4.47	9.92	143.82	4.54	109.05	21.39	111.31	15.60

FIGURE 4.53 – Portion du dataset OBSERVATION après transformation

4.7.2 Génération de la table de fait ClinicalFact :

En utilisant une table de fait sans valeurs mesurées, on peut simplifier la structure de données et faciliter la gestion des données historiques dans un contexte médical. Cette table a pour objectif regrouper les activités du patient revenant à une même date.

Citons qu'un patient a subi une consultation dans une date bien appropriée , a pris un médicament et a fait des analyses... Cette table peut nous donner une idée générale sur ces activités.

A partir de l'ensemble des dataframe qu'on a créé précédemment , on va utiliser la fonction `Merge()` en python pour joindre les datasets en utilisant les colonnes communes. Puisqu' on a divers types de jointures , dans notre cas , on va utiliser la jointure gauche (Voir Annexe 1 : Bibliothèques en Python) .

De ce fait , nous avons obtenu cette table :

	PatientUID	Date	Condition ID	Procedure ID	Encounter ID	ObservationID	Medication ID	Imminization ID
0	5cbc121b-cd71-4428-b8b7-31e53eba8184	2010-12-07	6	1	5	7990	0	0
1	5cbc121b-cd71-4428-b8b7-31e53eba8184	2010-12-07	6	1	372	7990	0	0
2	5cbc121b-cd71-4428-b8b7-31e53eba8184	2010-12-07	85	1	5	7990	0	0
3	5cbc121b-cd71-4428-b8b7-31e53eba8184	2010-12-07	85	1	372	7990	0	0
4	5cbc121b-cd71-4428-b8b7-31e53eba8184	2010-12-07	6	2	5	7990	0	0
...
80626	d8724548-d0e5-4ed3-a60c-6730e0d93e57	1978-08-21	0	0	0	0	14015	0
80627	ce356d64-2e4a-47b4-8a11-25ff968a1409	2015-02-13	0	0	0	0	14047	0
80628	ce356d64-2e4a-47b4-8a11-25ff968a1409	2015-02-13	0	0	0	0	14048	0
80629	3160b290-8c66-4a5c-a77b-cb2a3eab495d	2019-04-25	0	0	0	0	14061	0
80630	3160b290-8c66-4a5c-a77b-cb2a3eab495d	2019-04-25	0	0	0	0	14062	0

FIGURE 4.54 – Portion du dataset final de factClinical

On va réutiliser le même processus de transformation et de nettoyage réalisé au niveau des dimensions pour garantir la cohérence des données et améliorer leur qualité .

4.7.3 Chargement des données

La dernière étape de notre processus est le chargement des données dans Postgresql. On a commencé par créer des tables . Puis, on a utilisé la Commande `COPY()` afin de remplir les tables.

a. Problème et Solution :

Cependant, travaillant dans le concept de Big Data, nous devons être conscients des risques de latence qui pourraient avoir un impact négatif sur la performance de notre tableau de bord "Dashboard" par suite . Pour éviter cela, nous devons considérer les techniques de partitionnement des données comme une solution .

Le partitionnement est une technique de gestion de données utilisée dans les bases de données pour diviser une table volumineuse en parties plus petites et plus gérables appelées partitions. Le partitionnement permet de diviser des données en

fonction de certains critères, tels que la valeur d'une colonne ou d'un ensemble de colonnes, ce qui permet de réduire la quantité de données accédées lors d'une requête. Les partitions peuvent être stockées sur différents disques, ce qui peut améliorer les performances en répartissant la charge de lecture/écriture sur plusieurs disques simultanément.

Le partitionnement peut également faciliter la maintenance des tables, car il est plus facile de sauvegarder et de restaurer des partitions individuelles plutôt que la table entière. De plus, le partitionnement peut être utilisé pour répartir les données sur plusieurs serveurs, ce qui peut améliorer les performances en répartissant la charge de travail sur plusieurs machines.

b. Mise en place de la solution :

PostgreSQL[17] offre plusieurs méthodes de partitionnement, notamment le partitionnement natif PostgreSQL, le partitionnement avec PostgreSQL Foreign Data Wrapper (FDW) et le partitionnement avec des outils tiers. Le partitionnement natif PostgreSQL est une méthode intégrée de partitionnement prise en charge par PostgreSQL depuis la version 10.

Les partitions sont créées à l'aide de la clause `PARTITION BY` lors de la définition de la table, ce qui permet de spécifier les colonnes à utiliser pour partitionner les données. Et c'est le type qu'on va l'adapter .

Dans notre cas , nous utiliserons les requêtes le plus souvent sur les colonnes dates ou bien les colonnes patientUID , c'est pourquoi , on va appliquer le partitionnement sur ces deux colonnes. On a utilisé le partitionnement par intervalles pour les colonnes date et le partitionnement par hachage pour les colonnes de patientUID. Finalement , on a utilisé les clés primaires étant qu'un index pour améliorer les performances de la base de données. Cela permet d'accélérer les requêtes et assure une recherche plus rapide des données.

On cite alors l'exemple des partitions de "factClinical" dans PostgreSQL :

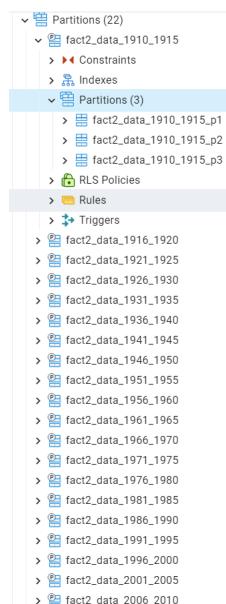


FIGURE 4.55 – Partitionnement de factClinical dans PostgreSQL

On vous cite ici la différence de temps de réponse avant et après le partitionnement :

Total rows: 1000 of 161262 Query complete 00:00:00.383

FIGURE 4.56 – Temps de réponse avant le partitionnement



Total rows: 1000 of 80631 Query complete 00:00:00.292

FIGURE 4.57 – Temps de réponse après le partitionnement

4.8 Conclusion :

Pour le département clinical, les données ont subit le processus ETL. Cela permet de faciliter l'analyse des données à des fins de recherche, la prise de décision et le traitement de données qui vont être réalisé après .On a décrit les étapes de mise en place de notre approche sur ce département dès les fichiers json éparpillés jusqu'à ce qu'on a créé une base de données concrète.

chapitre 5

Sprint 3 : Conception et mise en œuvre du domaine financial

5.1 Introduction :

Le département Financial gère toutes les transactions financières liées à la prestation de soins de santé, depuis la facturation des services jusqu'au traitement des demandes de paiement et des réclamations. Il permet également de suivre les remboursements, les contrats et les notifications liés à l'assurance maladie ainsi que de la communication d'informations entre les assureurs, les abonnés, les patients et d'autres parties prenantes.

5.2 Outil de gestion de base des données :

Postgresl sera considéré comme support de stockage des données pour le département Financial. Ces fonctionnalités nous permettront de stocker les données financières de manière efficace et de les gérer avec une grande flexibilité. En outre, l'utilisation de PostgreSQL est largement répandue dans le domaine de la santé, ce qui facilite l'interopérabilité et l'échange de données avec d'autres systèmes de santé.



FIGURE 5.58 – Logo de PostgrSQL

5.3 Outil d' ETL :

Pour le département Financial, nous avons décidé d'utiliser Python pour le processus ETL. Ce choix est motivé par sa flexibilité et sa compatibilité avec un large éventail de bibliothèques et de frameworks de traitement de données. En utilisant Python, nous serons en mesure d'automatiser et de simplifier le processus ETL pour le département Financial, ce qui nous permettra de gagner du temps et d'assurer la qualité des données.



FIGURE 5.59 – logo de Python

5.4 Conception du Domaine Financial :

Dans le cadre du département financial, nous allons créer un entrepôt de données qui rassemblera les données de remboursement, les informations sur les contrats, les paiements et autres données financières. Cette centralisation de données peut aider les informaticiens à effectuer des analyses et à prendre des décisions éclairées sur les activités financières de l'organisation. Cela peut également permettre de détecter les tendances et les anomalies dans les données financières. On cite ci-dessous le schéma du domaine Financial :

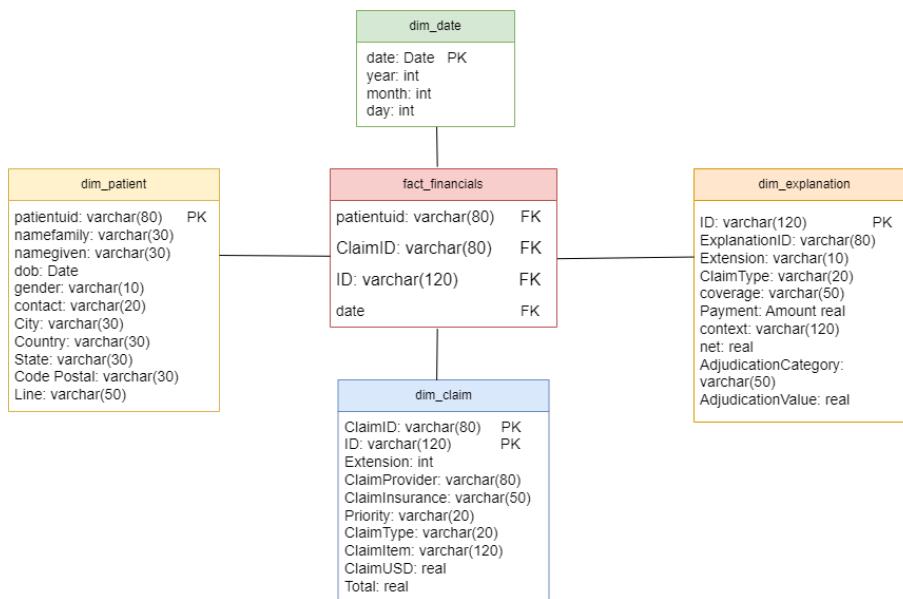


FIGURE 5.60 – Conception du domaine Financial

5.4.1 Modélisation :

Nous allons vous présenter les principales tables de dimension qui composent notre data warehouse dans le domaine financial, fournissant ainsi une base solide pour comprendre notre approche de gestion des données médicales relatives aux remboursements, facturations ...

Dimensions	Définition	Attributs
Patient	Informations démographiques et administratives sur un individu ou un animal recevant des soins ou d'autres services liés à la santé.	PatientUID : varchar(80) PK NameFamily : varchar(30) NameGiven : varchar(30) DoB : Date Gender : varchar(10) Contact : varchar Country : varchar(30) City : varchar(30) State : varchar(30) Code Postal : varchar(30)
ExplanationOf-Benefit	combine les informations de la demande et de la réponse à la demande, en supprimant les informations propriétaires des fournisseurs ou des payeurs, dans un modèle d'information unifié adapté à une utilisation pour : les rapports de patients, le transfert d'informations vers un système de dossier médical du patient et la prise en charge complète des informations d'adjudication et de demande échangées avec des organismes de réglementation et d'analyse, ainsi qu'avec d'autres parties de l'organisation du fournisseur.	ID : varchar(120) PK ExplanationID : varchar(80) ClaimType : varchar(20) ClaimType : varchar(20) coverage : varchar(50) Payment Amount : real context : varchar(120) net : real AdjudicationCategory : varchar(50) AdjudicationValue : real

Claim	Les fournisseurs et les payeurs, notamment les assureurs, utilisent la demande pour échanger des informations financières et cliniques liées à la prestation de services de soins de santé avec les payeurs, ainsi que pour la production de rapports destinés aux organismes de réglementation et aux entreprises fournissant des analyses de données.	ClaimID : varchar(80) PK ID : varchar(120) PK Extension : int ClaimProvider : varchar(80) ClaimInsurance : varchar(50) Priority : varchar(20) ClaimType : varchar(20) ClaimItem : varchar(120) ClaimUSD : real
Date	Une dimension qui sert à l'historisation des données .	Date : Date Year : int Month : int Day : int

TABLE 5.6 – Conception du domaine Financial

5.5 Mise en place d'un processus ETL :

Notez bien : Le problème rencontré qui concerne la complexité de la manipulation des fichiers json de HL7 FHIR a la même solution proposée citée dans le domaine “Individuals” et “clinical” .

5.5.1 Processus ETL pour les dimensions :

Notez bien : Étant donnée que nous travaillons dans une architecture Data mesh on peut accéder au données des autres domaines grâce à la plateforme . Ainsi , on va utiliser les données du table Patient du domaine Clinical puisque on a le même besoin .

Extraction :

Pour extraire les données dans le département des finances, nous pouvons utiliser la bibliothèque de ressources FHIR de Python(Dans l'annexe 1, nous avons parlé de cette bibliothèque), qui fournit une interface pour interagir avec les ressources FHIR.

Pour extraire les données, nous devons spécifier la ressource appropriée à chaque fois. Par exemple, si nous voulons extraire des informations sur les factures de soins de santé, nous devons spécifier la ressource "Claim" dans notre script Python. En utilisant ces ressources, nous pouvons extraire les données pertinentes et les stocker dans une base de données ou les utiliser pour des analyses financières ultérieures.

- ♣ ExplanationOfBenefit : Voici un extrait du code pour la dimension ExplanationOfBenefit qui permet d'extraire les données des fichiers json de type HL7 FHIR après avoir distinguer le Bundle et les ressources de type "ExplanationOfBenefit" :

```

for j in range(len(resExplanation)):
    oneExp = ExplanationOfBenefit.parse_obj(resExplanation[j])
    for i in range(len(resExplanation[j].item)):
        if resExplanation[j].item[i].adjudication:
            for cpt in range(len(resExplanation[j].item[i].adjudication)):
                ids.append(oneExp.identifier[0].value)
                extension.append('+'+str(i+1)+ '-' +str(cpt+1))
                type.append(oneExp.type.coding[0].code)
                coverage.append(oneExp.insurance[0].coverage.display)
                TotalAmount.append(oneExp.total[0].amount.value)
                paymentAmount.append(oneExp.payment.amount.value)
                date.append(oneExp.created)
                condition.append(oneExp.item[i].productOrService.text)
                adjudicationCategory.append(oneExp.item[i].adjudication[cpt].category.coding[0].display)
                if oneExp.item[i].adjudication[cpt].amount :
                    adjudicationValue.append(oneExp.item[i].adjudication[cpt].amount.value)
                else :
                    adjudicationValue.append(np.nan)
                if resExplanation[j].item[i].net:
                    net.append(str(resExplanation[j].item[i].net.value))
                else:
                    net.append(np.nan)

            else :
                ids.append(oneExp.identifier[0].value)
                extension.append('+'+str(i+1)+ '-0')
                type.append(oneExp.type.coding[0].code)
                coverage.append(oneExp.insurance[0].coverage.display)
                TotalAmount.append(oneExp.total[0].amount.value)
                paymentAmount.append(oneExp.payment.amount.value)
                date.append(oneExp.created)
                condition.append(oneExp.item[i].productOrService.text)
                adjudicationCategory.append(np.nan)
                adjudicationValue.append(np.nan)
                if resExplanation[j].item[i].net:
                    net.append(str(resExplanation[j].item[i].net.value))
                else:
                    net.append(np.nan)

```

FIGURE 5.61 – Partie du code d'extraction du dimension ExplanationOfBenefit

- ♣ Claim : Voici un extrait du code pour la dimension Claim qui permet d'extraire les données des fichiers json de type HL7 FHIR après avoir distinguer le Bundle et après distiguier les ressources de type "Claim" :

```

for j in range(len(resClaims)):
    oneClaim = Claim.parse_obj(resClaims[j])
    # Inner Loop over claim items:
    for i in range(len(resClaims[j].item)):
        ids.append(oneClaim.id)
        extension.append('+'+str(i+1))
        claimProvider.append(oneClaim.provider.display)
        claimInsurance.append(oneClaim.insurance[0].coverage.display)
        claimDate.append(str(oneClaim.billablePeriod.start.date()))
        claimType.append(oneClaim.type.coding[0].code)
        claimItem.append(resClaims[j].item[i].productOrService.text)
        TotalAmount.append(oneClaim.total.value)
        priority.append(oneClaim.priority.coding[0].code)
        if resClaims[j].item[i].net:
            claimUSD.append(str(resClaims[j].item[i].net.value))
        else:
            claimUSD.append('None')

```

FIGURE 5.62 – Partie du code d'extraction du dimension Claim

Transformation :

L'action qu'on va mener plus tard sur les données lors de leur chargement sera l'insertion ou la mise à jour, cette action nécessite forcément la présence d'une clé primaire dans le schéma des données. Dans ce contexte , la colonne "ClaimUID" identifie chaque demande de manière unique, mais il peut également y avoir une colonne d'extension pour fournir des informations supplémentaires. Au lieu de conserver deux colonnes d'identifiant unique, il est recommandé de combiner les colonnes de ClaimUID et d'extension pour créer une seule colonne "ID" qui servira de clé primaire dans notre base de données. Il est également important de faire un nettoyage des données pour éliminer les erreurs ou les données redondantes. Le nettoyage des données peut inclure la suppression des enregistrements en double, la vérification de l'intégrité des données, la normalisation des données et l'élimination des données inutiles ou obsolètes.

♣ ExplanatoryBenefit :

Voici une portion du Dataset EXPLANATION après transformation :

3	80ea06f4-881b-4d74-a443-24dc4fd2acaf1-0	80ea06f4-881b-4d74-a443-24dc4fd2acaf	1-0	institutional	Humana	0.000	1977-02-21 11:52:41+00:00	General examination of patient (procedure)	NaN	NaN	NaN	5cbc121b-cd71-4428-bb87-31e53eba8184
4	80ea06f4-881b-4d74-a443-24dc4fd2acaf2-0	80ea06f4-881b-4d74-a443-24dc4fd2acaf	2-0	institutional	Humana	0.000	1977-02-21 11:52:41+00:00	Body mass index 30+ - obesity (finding)	NaN	NaN	NaN	5cbc121b-cd71-4428-bb87-31e53eba8184
...
364413	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6-2-2	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6	2-2	institutional	UnitedHealthcare	112.416	2019-07-04 14:44:43+00:00	Influenza, seasonal, injectable, preservative ...	140.52	Line Provider Payment Amount	112.416	c588a992-d308-4916-aed5-3894f7e5f6e3
364414	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6-2-3	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6	2-3	institutional	UnitedHealthcare	112.416	2019-07-04 14:44:43+00:00	Influenza, seasonal, injectable, preservative ...	140.52	Line Submitted Charge Amount	140.520	c588a992-d308-4916-aed5-3894f7e5f6e3
364415	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6-2-4	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6	2-4	institutional	UnitedHealthcare	112.416	2019-07-04 14:44:43+00:00	Influenza, seasonal, injectable, preservative ...	140.52	Line Allowed Charge Amount	140.520	c588a992-d308-4916-aed5-3894f7e5f6e3
364416	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6-2-5	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6	2-5	institutional	UnitedHealthcare	112.416	2019-07-04 14:44:43+00:00	Influenza, seasonal, injectable, preservative ...	140.52	Line Beneficiary Part B Deductible Amount	0.000	c588a992-d308-4916-aed5-3894f7e5f6e3
364417	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6-2-6	fd8b2c08-9dcf-4ff2-841d-4c9d876e6cf6	2-6	institutional	UnitedHealthcare	112.416	2019-07-04 14:44:43+00:00	Influenza, seasonal, injectable, preservative ...	140.52	Line Processing Indicator Code	NaN	c588a992-d308-4916-aed5-3894f7e5f6e3

FIGURE 5.63 – Portion du Dataset EXPLANATION

♣ Claim :

Voici une portion du Dataset CLAIM après transformation :

ClaimID	Extension	ClaimProvider	ClaimInsurance	ClaimDate	Priority	ClaimType	ClaimItem	ClaimUSD	Total	PatientUID
55d10840-4850-43dd-bab2-334ae3e68839	1	MORTON HOSPITAL	Humana	1965-11-15	normal	institutional	Cardiac Arrest	None	129.16	5cbc121b-cd71-4428-bb87-31e53eba8184
55d10840-4850-43dd-bab2-334ae3e68839	2	MORTON HOSPITAL	Humana	1965-11-15	normal	institutional	Cardiac Arrest	None	129.16	5cbc121b-cd71-4428-bb87-31e53eba8184
55d10840-4850-43dd-bab2-334ae3e68839	3	MORTON HOSPITAL	Humana	1965-11-15	normal	institutional	History of cardiac arrest (situation)	None	129.16	5cbc121b-cd71-4428-bb87-31e53eba8184
80ea06f4-881b-4d74-4443-24dc4fd2acaf	1	PCP116891	Humana	1977-02-21	normal	institutional	General examination of patient (procedure)	None	129.16	5cbc121b-cd71-4428-bb87-31e53eba8184
80ea06f4-881b-4d74-4443-24dc4fd2acaf	2	PCP116891	Humana	1977-02-21	normal	institutional	Body mass index 30+ - obesity (finding)	None	129.16	5cbc121b-cd71-4428-bb87-31e53eba8184

FIGURE 5.64 – Portion du Dataset CLAIM

5.5.2 Génération de la table de fait FinancialFact :

Pour construire notre table de fait à partir de l'ensemble des data frames qu'on a créé précédemment .

On a fait recours à la fonction Merge() en python avec sa jointure externe qui inclut tous les enregistrements de chaque ensemble de données, même s'il n'y a pas de correspondance avec l'autre ensemble, garantissant ainsi l'inclusion de toutes les informations disponibles.

```
df1=EXPLANATION.merge(CLAIM,on=["PatientUID","date","id","ClaimType"],
how='outer',suffixes=('_E','_C'))
```

Le nettoyage des données pour la table des faits est crucial pour assurer l'exactitude et la fiabilité des données qui y sont stockées. Tout d'abord, il est important d'éliminer les données en double. Il est également essentiel de normaliser les données en utilisant des unités de mesure standardisées et des codes de catégorie pour assurer la cohérence des données. Enfin, il est important d'éliminer les données obsolètes ou inutiles qui peuvent encombrer la table et affecter les performances de la base de données.

5.5.3 Chargement des données :

La dernière étape de notre processus est le chargement des données dans Postgresql. Vu la masse des données qu'on a dans ce département, on a fait recours au partitionnement pour garantir la performance des requêtes et l'optimisation de l'espace disque. On a appliqué un partitionnement par hachage pour le PatientUID et un sous partitionnement par intervalle pour la date. Une fois que le partitionnement est défini, nous pouvons procéder au chargement des données dans PostgreSQL en utilisant pgAdmin où nous avons utilisé la commande **COPY()**. Voici un extrait du factFinancial dans PostgreSQL

	id_e	id_c	id	date	patientuid
1	551010840-4850-43ddbab2-334ee3e8899-1-0	551010840-4850-43ddbab2-334ee3e8899-1	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
2	551010840-4850-43ddbab2-334ee3e8899-1-0	551010840-4850-43ddbab2-334ee3e8899-2	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
3	551010840-4850-43ddbab2-334ee3e8899-1-0	551010840-4850-43ddbab2-334ee3e8899-3	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
4	551010840-4850-43ddbab2-334ee3e8899-2-0	551010840-4850-43ddbab2-334ee3e8899-1	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
5	551010840-4850-43ddbab2-334ee3e8899-2-0	551010840-4850-43ddbab2-334ee3e8899-2	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
6	551010840-4850-43ddbab2-334ee3e8899-2-0	551010840-4850-43ddbab2-334ee3e8899-3	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
7	551010840-4850-43ddbab2-334ee3e8899-3-0	551010840-4850-43ddbab2-334ee3e8899-0	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
8	551010840-4850-43ddbab2-334ee3e8899-3-0	551010840-4850-43ddbab2-334ee3e8899-1	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
9	551010840-4850-43ddbab2-334ee3e8899-3-0	551010840-4850-43ddbab2-334ee3e8899-3	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
10	80ea0f44-881b-4f71-a443-24ec4fd2aaef-1-0	80ea0f44-881b-4f71-a443-24ec4fd2aaef-1	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
11	80ea0f44-881b-4f71-a443-24ec4fd2aaef-1-0	80ea0f44-881b-4f74-a443-24ec4fd2aaef-2	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
12	80ea0f44-881b-4f71-a443-24ec4fd2aaef-2-0	80ea0f44-881b-4f74-a443-24ec4fd2aaef-1	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
13	80ea0f44-881b-4f71-a443-24ec4fd2aaef-2-0	80ea0f44-881b-4f74-a443-24ec4fd2aaef-2	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
14	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-1-0	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-1	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
15	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-1-0	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-2	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
16	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-1-0	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-3	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
17	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-2-0	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-1	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7
18	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-2-0	1ca3babcd-f3fe-4938-8a1d-e7c1ac4f9658-2	Sdb121b-cd71-4429-b8b7	1965-11-15	Sdb121b-cd71-4429-b8b7

FIGURE 5.65 – Portion de factFinancial dans PostgreSQL

5.6 Conclusion :

Dans le département Financial, nous avons travaillé sur les données de demandes de remboursement de soins de santé pour aider à comprendre les coûts associés à la prestation de services de santé. On a décrit les étapes de mise en place de notre approche sur ce département dès les fichiers json éparsillés jusqu'à ce qu'on a créé une base de données fiable et précise qui peut être utilisée pour des analyses et des rapports financiers dans le domaine de la santé.

chapitre 6

Sprint 4 : Développement d'une application web produite par le département Clinical

6.1 Introduction :

Dans ce chapitre, nous allons décrire le processus de développement de notre application web monolithique , en mettant l'accent sur la manière dont nous avons exposé les API de notre base de données PostgreSQL et développé un Dashboard pour transformer les données en informations exploitables et fournir une vue d'ensemble utiles aux praticiens et aux décideurs dans le domaine de la santé.

6.2 Exposition des API :

L'exposition des API est un aspect important lors de la conception d'une application web, car elle permet à d'autres applications ou systèmes de communiquer avec votre application en utilisant des interfaces standardisées. Nous allons décrire la manière dont les API sont exposées dans notre application web développée avec Angular et Spring Boot.

6.2.1 API RESTful :

Une API RESTful (Representational State Transfer) utilise le protocole HTTP pour interagir avec des clients. Elle se base sur les principes de l'architecture REST qui établit un ensemble de règles pour concevoir des systèmes d'information évolutifs, extensibles et aisément maintenables. Les caractéristiques principales d'une API RESTful sont :

- ♣ Les méthodes HTTP standards (GET, POST, PUT, DELETE) sont utilisées pour réaliser des actions sur les ressources.
- ♣ Utilisation d'URLs (Uniform Resource Locators) pour identifier les ressources et les opérations à effectuer.

- ♣ Utilisation de formats de données standard comme JSON ou XML pour représenter les données transmises entre le client et le serveur.
- ♣ Absence de stockage d'état sur le serveur : chaque requête du client doit inclure toutes les informations nécessaires pour que le serveur puisse la traiter.

L'application de ces principes permet à une API RESTful de proposer une interface simple et cohérente pour interagir avec une application, et facilite l'intégration avec d'autres applications tierces.

6.2.2 Jhipster :

JHipster[18] est un générateur d'applications Web qui utilise Spring Boot et Angular pour créer rapidement des applications modernes et robustes en fournissant une architecture bien structurée, une configuration de sécurité, une gestion des utilisateurs et des API RESTful. Cela permet aux développeurs de se concentrer sur le développement de fonctionnalités plutôt que sur la configuration de l'infrastructure sous-jacente.

D'après les statistiques du mois mai 2023 , 78,3% des utilisateurs ont évalué la qualité du code généré par JHipster comme étant de 8 sur 10 ou plus ! Et voici une figure qui illustre le nombre des étoiles dans github et le nombre des téléchargements .



FIGURE 6.66 – Popularité de jhipster

6.2.3 Réalisation :

La création d'une application Web nécessite une bonne planification et une bonne compréhension des besoins de l'entreprise ou de l'utilisateur final. C'est là que le langage de description de domaine spécifique entre en jeu et nous allons décrire le processus de création des fichiers JDL pour notre projet, ainsi que les entités et les relations que nous avons définies en utilisant ce langage.

JDL :[19]

Le "JDL" , signifie "JHipster Domain Language", est un langage de description de domaine spécifique (DSL) utilisé par JHipster pour décrire les entités et les relations du modèle de données de l'application, ainsi que les règles de validation et les autorisations.

Utilisation des JDL :

Il y a deux cas de création d'une application en utilisant les JDL (JHipster Domain Language) :

- ♣ Création à partir d'un fichier JDL existant : Dans ce cas, vous devez avoir un fichier JDL contenant la description de votre application, puis exécuter la commande "`jhipster import-jdl <chemin du fichier JDL>`". JHipster va ensuite générer le code de votre application en utilisant les informations fournies dans le fichier JDL.
- ♣ Création à partir de zéro : Dans ce cas, vous pouvez utiliser la commande "`jhipster`" pour créer une nouvelle application vierge. Vous serez ensuite invité à répondre à plusieurs questions pour configurer votre application. Vous pouvez également utiliser l'option "`skip-server`" pour générer uniquement le frontend de l'application.

Dans notre cas, nous avons créé un fichier JDL décrivant les entités, les champs et les relations entre les entités de notre application, puis nous avons utilisé cet outil pour générer les fichiers nécessaires à notre application JHipster. Cette approche nous a permis de gagner du temps en évitant d'avoir à écrire manuellement tout le code pour notre application.

Création et configuration des jdl des domaines :

La création de fichiers JDL pour chaque domaine est une étape importante dans la construction d'une architecture Data Mesh. Chaque domaine doit être traité comme une entité distincte et autonome, avec ses propres besoins en matière de données et de traitement. En utilisant JHipster et le JDL, nous pouvons décrire chaque domaine en termes d'entités, de relations et de contraintes de validation. Les paramètres de configuration incluent notamment le nom de l'application, le type d'application (monolithique dans notre cas), l'utilisation de Swagger pour la génération de code, le préfixe des noms de classes, le nom de package, les types de base de données pour le développement et la production, l'outil de construction (ici Maven). Enfin, l'option "`entities *`" signifie que toutes les entités seront générées automatiquement à partir de la base de données.

Intégration de la base de données dans JHipster :

La première étape de l'intégration a été de configurer JHipster pour utiliser PostgreSQL en tant que base de données. Nous avons effectué cette configuration en utilisant les options fournies par JHipster pour générer les fichiers de configuration appropriés pour Spring Boot.

Ensuite, nous avons créé le schéma de la base de données en utilisant les entités générées par JHipster. Cela nous a permis de définir les tables, les colonnes, les clés primaires et étrangères, ainsi que les contraintes de vérification.

Après avoir créé le schéma, nous procédons à l'exécution des migrations de base de données pour créer toutes les tables et les relations correspondantes. Nous avons également ajouté des données de test pour vérifier que la base de données fonctionne correctement.

Dans le cadre de la migration de ces données, nous devons déclarer les informations de connexion à la base de données PostgreSQL dans différents fichiers comme pom.xml, application-prod.xml, application-dev.xml et postgresql.xml. Ces fichiers contiennent des informations telles que les noms d'utilisateur et les mots de passe nécessaires pour se connecter à la base de données. Cette étape est essentielle pour que notre application puisse correctement communiquer avec la base de données et que les différentes fonctions de l'application puissent fonctionner correctement.

Enfin, nous avons utilisé la fonctionnalité fournie par JHipster pour interagir avec la base de données. Nous avons créé des référentiels pour chaque entité, ce qui nous a permis d'effectuer des opérations CRUD sur la base de données.

En résumé, l'intégration de la base de données PostgreSQL dans JHipster nous permet de configurer, créer et manipuler la base de données, ce qui nous permet de nous concentrer sur le développement d'API et de tableaux de bord. On cite comme exemple une vue des données de l'entité Patient :

ID	Patient UID	Name Family	Name Given	Birthday	Gender	Contact	Line	City	Country	State	Postalcode	Actions
1	SOC0705-071-4408-8807-31x3a0a8f84	Brielle498	Aaron697	10 Dec 1945	male	550-877-3719	934 Braeck Bypass	Taunton	US	Massachusetts	2710,0	View Edit Delete
2	a80C0705-0500-4C09-8e23-0f980247f29	Stederman142	Aaron697	29 Mar 1946	male	550-219-2054	378 Kregg Lodge	Westford	US	Massachusetts	2710,0	View Edit Delete
3	31919303-0400-4479-930-e0f02ca3f39	Kurawitz899	Abby752	24 Oct 2002	female	550-190-9152	892 Hoppe Annex	Wellesfield	US	Massachusetts	1880,0	View Edit Delete
4	07818310-4211-4039-8f6e-407718fb7e4	Connell942	Alecia832	12 Dec 1949	male	550-728-3242	838 Braeck Union Suite 44	Hornbridge	US	Massachusetts	2710,0	View Edit Delete
5	b4260102-4273-4839-9007-93170502851	Heiter142	Alannah100	16 Apr 1997	male	550-820-7962	834 Hansen Run	Somerville	US	Massachusetts	2710,0	View Edit Delete
6	528F50-705-1403-8253-503249154	Cory187	Ashley181	24 Dec 1997	male	550-281-8550	441 Zenith Union Unit 91	Milton	US	Massachusetts	2710,0	View Edit Delete
7	34640295-5402-4026-8726-0787931a074	Sheldene52	Adam181	29 Nov 2021	male	550-470-8788	173 Abbie Heights	Ludlow	US	Massachusetts	2710,0	View Edit Delete
8	65835198-8602-4571-aef1-5fc6ebe7e74	Gulgowski15	Adele184	14 Nov 1999	female	550-317-2203	721 Iron Works Unit 31	Plymouth	US	Massachusetts	2340,0	View Edit Delete
9	04303532-1400-4203-8776-2440c23995	Ovelette54	Aldo1777	3 Feb 2003	male	550-405-4440	721 Blanda Light Unit 95	Roxbury	US	Massachusetts	2710,0	View Edit Delete
10	50122075-5129-4024-8705-16730f88820	Turcotte310	Aldo1780	4 Dec 1995	male	550-480-5160	217 Will Spur Suite 31	Amherst	US	Massachusetts	2710,0	View Edit Delete
11	3966b610-1548-4284-8761-16e07379598	Williamson79	Adolph80	17 Feb 1993	male	550-367-1730	506 Golder Parade	Brookline	US	Massachusetts	2210,0	View Edit Delete
12	707877C-064-4596-9204-DE24a556b	Ankrum277	Aldrin871	2 Mar 1998	female	550-126-2866	547 Flicker Passage Unit 33	Hopkinton	US	Massachusetts	1747,0	View Edit Delete
13	23127751-9443-4149-837c-5fcdca48918	Jenkin74	Agneta234	8 Aug 1997	female	550-239-5872	802 Pleasant Stuyvesant	Westford	US	Massachusetts	2710,0	View Edit Delete
14	609f9e10-5149-4049-b659-40770597a72	Gordon134	Ajani159	26 Dec 1995	male	550-760-4142	176 Ivan Course Suite 77	Worchester	US	Massachusetts	1545,0	View Edit Delete
15	87994400-1E14-4583-A208-02377174781	Nader710	Alyssa085	21 Apr 1991	male	550-380-1300	323 Bernard Spur	Boston	US	Massachusetts	2108,0	View Edit Delete
16	6329710-0564-4404-9455-07837174f13	Rummel78	Alik1035	14 Jun 1999	female	550-252-43394	1058 Carroll Drive Apt 22	Wayne	US	Massachusetts	2710,0	View Edit Delete
17	82f2620-0503-401-8C2-147620447	Rosenbaum74	Aliza23	15 Jun 1999	male	550-903-9162	528 Langdon Park	Bedford	US	Massachusetts	1915,0	View Edit Delete

FIGURE 6.67 – Entité Patient dans le back-end

On illustre de même la liste des schéma existante :



FIGURE 6.68 – liste des schémas

6.2.4 Documentation des APIs :

La documentation Swagger permet de documenter facilement l'API REST que nous avons créée avec JHipster. Pour mettre en place la documentation Swagger, nous avons ajouté la dépendance "springfox-swagger2" dans le fichier "pom.xml" de notre projet Maven. Ensuite, nous avons créé une classe de configuration Swagger où nous avons configuré Swagger pour scanner automatiquement nos contrôleurs REST et générer la documentation Swagger correspondante. Nous avons également ajouté des annotations Swagger à nos contrôleurs REST pour spécifier les informations supplémentaires à inclure dans la documentation Swagger. Une fois la configuration terminée, nous avons accédé à la documentation Swagger à l'aide de l'URL "/swagger-ui.html" dans notre navigateur web. On cite par exemple la documentation de l'API dans Swagger de l'entité Procedure :

The screenshot shows the Swagger UI interface for the 'Clinical API'. At the top, it displays the title 'Clinical API 0.0.1 [OAS]'. Below the title, there's a dropdown menu labeled 'Select a definition' with 'Clinical (default)' selected. The main content area is titled 'procedure-resource'. It lists several API endpoints with their HTTP methods and URLs:

- GET /api/procedures/{id}**
- PUT /api/procedures/{id}**
- DELETE /api/procedures/{id}**
- PATCH /api/procedures/{id}**
- GET /api/procedures**
- POST /api/procedures**

FIGURE 6.69 – Exemple de documentation de l'API dans Swagger

6.2.5 Sécurité des APIs :

Spring Security est une bibliothèque de sécurité puissante et largement utilisée pour les applications Java. Elle fournit des fonctionnalités de sécurité telles que l'authentification, l'autorisation, la gestion des sessions et la protection contre les attaques de sécurité telles que les injections SQL et les attaques par déni de service. Dans notre projet, nous avons utilisé Spring Security pour sécuriser notre API JHipster. Nous avons configuré des filtres de sécurité pour protéger les points de terminaison de l'API contre les utilisateurs non autorisés. Nous avons également configuré l'authentification et l'autorisation en utilisant Spring Security.

Pour l'authentification, nous avons utilisé l'authentification basée sur les jetons JWT (JSON Web Token). Cela nous a permis de fournir une authentification sécurisée et sans état pour nos utilisateurs.

Pour l'autorisation, nous avons configuré des rôles d'utilisateur pour contrôler l'accès aux différents points de terminaison de l'API. Nous avons également configuré des autorisations basées sur les rôles pour permettre des actions spécifiques pour des utilisateurs avec des rôles spécifiques.

En utilisant Spring Security, nous avons pu ajouter une couche de sécurité robuste à notre application JHipster, en protégeant les données sensibles et en restreignant l'accès aux fonctionnalités de l'API en fonction des rôles de l'utilisateur.

6.2.6 Performance de l'API :

Dans une application , l'affichage d'une grande quantité de données peut affecter les performances de l'application, et la pagination permet de limiter la quantité de données extraites en une seule fois, réduisant ainsi le temps de chargement et améliorant les performances de l'application. JHipster intègre automatiquement la pagination dans l'API générée en créant des entités à l'aide de JDL, ce qui facilite le développement d'applications axées sur la logique métier. En tant que telle, la pagination est une fonctionnalité clé de toute application JHipster qui gère de grandes quantités de données.

Et voici un exemple de la pagination dans l'entité Encounter :

23702	Prenatal visit	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	17 Jun 2019	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23652	Patient encounter procedure	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	13 Oct 2011	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23694	Prenatal visit	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	25 Sep 2017	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23703	Prenatal visit	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	15 Jul 2019	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23680	Prenatal visit	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	1 Aug 2018	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23678	Prenatal visit	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	6 Jun 2016	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23664	Prenatal visit	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	19 Jan 2015	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23684	Obstetric emergency hospital admission	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	17 Oct 2016	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23660	Consultation for treatment	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	6 Oct 2013	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23661	Patient encounter procedure	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	29 Sep 2014	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23688	Prenatal visit	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	24 Apr 2017	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete
23695	Prenatal initial visit	HALLMARK HEALTH SYSTEM	Dr Renato359 Jenkins714	27 Feb 2017	00807762-6d48-43f6-b6fb-fe4c8abae329	View Edit Delete

FIGURE 6.70 – Exemple de pagination

6.3 Les outils de développement :

Angular et Spring Boot sont deux technologies très populaires pour la création d'applications web monolithes. Angular est un framework de développement front-end basé sur TypeScript, tandis que Spring Boot est un framework de développement back-end basé sur Java. Ensemble, ces technologies peuvent être utilisées pour créer une application web monolithique robuste et performante.

6.3.1 Angular :

Angular [20] est un framework de développement front-end open-source créé par Google pour développer le front-end des applications web. Les développeurs peuvent utiliser Angular pour créer des composants, des directives et des services responsables de l'affichage des données et de la gestion des interactions utilisateur. Angular offre une architecture orientée composants, une liaison de données bidirectionnelle, des fonctionnalités de routage et d'injection de dépendances, ainsi que des fonctionnalités intégrées pour faciliter les tests unitaires et d'intégration. Les données sont généralement récupérées à partir du back-end via des requêtes HTTP.

6.3.2 Spring Boot :

Spring Boot[21] est un framework de développement open source basé sur Spring qui simplifie la construction d'applications Web backend. Il fournit des fonctionnalités telles que la configuration automatique, les dépendances intégrées, le serveur Web intégré, la gestion centralisée des erreurs et le module Actuate pour surveiller et gérer les applications. Avec sa facilité d'utilisation et sa capacité à réduire la quantité de configuration requise, Spring Boot est populaire pour développer des API RESTful qui manipulent les données et gèrent la sécurité, les sessions et les utilisateurs. Par conséquent, il est utilisé pour développer la partie backend de l'application Web, en association avec la technologie front-end Angular.

6.3.3 conclusion :

Dans une application web monolithique, le front-end et le back-end sont développés et déployés comme une seule application. Les avantages de cette approche incluent un déploiement et une gestion simplifiés des applications, des performances améliorées grâce à la communication entre le front-end et le back-end, et une meilleure cohérence et sécurité des applications.

6.4 Développement des interfaces :

6.4.1 Introduction :

Après avoir préparé les données et exposé leurs APIs , nous allons mettre l'accent dans cette partie sur les interfaces graphiques développées.

6.4.2 Objectif de l'application web :

Dans le domaine de la santé, le temps presse et les professionnels de santé ont besoin d'identifier rapidement les informations essentielles pour les soins des patients. L'objectif

de notre application web est d'afficher les données de manière claire et concise donc cet outil peut être précieux pour améliorer le processus de prise de décision et les résultats des soins des patients.

6.4.3 Les utilisateurs de l'application web :

L'application web que nous avons développée a été conçue pour répondre aux besoins des utilisateurs . Dans notre cas , pour accéder aux données et au tableau de bord l'utilisateur peut s'identifier en tant que :

- ♣ Administrateur : Les administrateurs sont responsables de la gestion des comptes d'utilisateurs et contrôlent l'accès aux ressources. Ils s'assurent également que la gestion des données est conforme au règlement.
- ♣ Professionnels de santé : Ils peuvent accéder aux ressources et gérer les informations sur les patients. Ils peuvent ajouter, supprimer et mettre à jour des informations médicales pour garantir des soins de qualité.

La figure suivante présente la page d'accueil pour l'authentification :

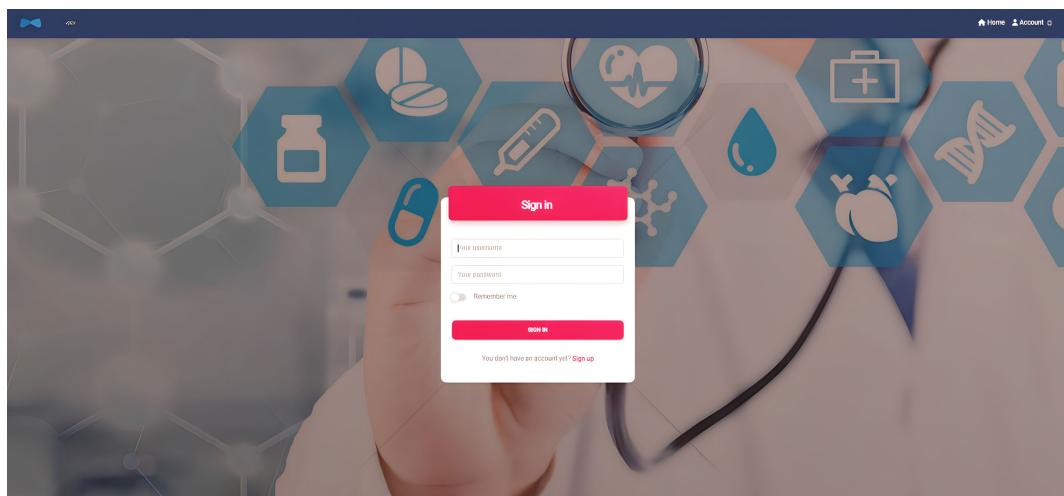


FIGURE 6.71 – Authentification

D'après cette fenêtre , on peut créer un nouveau compte d'utilisateur en remplissant ce formulaire :

The registration form consists of several input fields:

- Username
- Email
- New password
- Password strength (with a scale from 1 to 10)
- New password confirmation
- Confirm the new password

At the bottom right is a pink "register" button.

FIGURE 6.72 – Inscription

6.4.4 Les ressources dans l'application Clinical :

Une fois authentifié, les professionnels de santé peuvent accéder aux données et les modifier pour maintenir les dossiers des patients complets et à la page.
Cette figure présente l'ensemble des tables auxquelles les professionnels peuvent accéder :

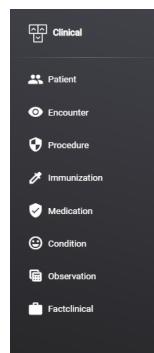


FIGURE 6.73 – Ensemble des tables existantes

En accédant à chacune des tables , on peut ajouter , modifier et supprimer les données. On montre dans la figure ci-dessous un exemple de table des encounters .

The screenshot shows the following views:

- Create or edit a Encounter**: A form with fields for ID (radio buttons), Encounters Text, Encounter Location, Encounter Provider, Date, and Patient UID. Buttons: CANCEL, SAVE.
- Encounter**: A list view showing an entry with ID 1, Encounters Text "Cardiac Arrest", Encounter Location "MORTON HOSPITAL", Encounter Provider "Dr. Bennett146 Breitenberg711", Date "15 Nov 1965", and Patient UID "5c6c121b-cd71-4428-b8b7-31e53eba8184". Buttons: BACK, EXIT.
- Create or edit a Encounter**: A second instance of the creation/edit form, identical to the first one.

FIGURE 6.74 – CRUD dans la table Encounter

Ces tables comportent également des filtrations afin de faciliter la recherche. Par Exemple , en accédant à la table des patient , le médecins peut faire des recherche en filtrant les données selon le nom , le prénom , la date de naissance et le genre .Cela permet d'effectuer une recherche rapide et identifier le "patientUID".

ID	Patient UID	Name Family	Name Given	Birthdate	Gender	Contact	Line	City	Country	State	Postcode
974	4f394d72-1828-459c-b9a3-0c29327ab0f5	Graham902	Ronnie7	14 May 2015	male	555-217-3863	723 Cassin Grove Apt 13	Arlington	US	Massachusetts	2231.0
1026	a6077197-e501-4269-8666-120ff1e75978	Owen916	Shelby741	22 Jun 1958	male	555-678-2299	324 Pfannerstill Frontage road Unit 61	Arlington	US	Massachusetts	2231.0
187	6605daac-4603-4674-b79f-096aa345f22d	Spirka222	Cath149	4 Apr 1973	female	555-208-8033	818 Herrinton Mill	Acton	US	Massachusetts	2231.0

FIGURE 6.75 – La recherche dans la table des patients

Nous avons également la table FactClinicals, qui contient des données historisées où toutes les entités sont liées via une date précise et un "patientUID". Dans ce tableau, autre que le filtrage des données, chaque champ est accessible en détail. Ce processus permet aux professionnels de la santé de consulter l'état d'un patient au fil de temps.
Cette figure illustre FactClinicals :

ID	Patient ID	Encounter ID	Observation ID	Procedure ID	Medication ID	Condition ID	Date
51	5dc1210b-cf71-4438-8d87-31e55ed8d194	16	7997	0	0	1	21 May 2016
52	5dc1210b-cf71-4438-8d87-31e55ed8d194	16	7997	0	0	1	21 May 2016
53	5dc1210b-cf71-4438-8d87-31e55ed8d194	380	7997	0	0	1	21 May 2016
54	5dc1210b-cf71-4438-8d87-31e55ed8d194	380	7997	0	0	1	21 May 2016
55	5dc1210b-cf71-4438-8d87-31e55ed8d194	16	7997	0	0	1	21 May 2016

FIGURE 6.76 – La table FactClinicals

En cliquant sur l'un des champs, vous serez redirigé sur une autre page, sur laquelle vous retrouverez plus d'informations sur le champs sélectionné. On présente un exemple de détails de l'entité Procedure :

Procedure

ID
10705

Procedure Text
Renal dialysis (procedure)

Date
16 Sep 2019

Patient UID
b0f49c80-b59b-4df6-8292-40ce8b8f8612

[← BACK](#) [EDIT](#)

FIGURE 6.77 – Détails du champs ProcedureID dans FactClinicals

6.4.5 Tableau de bord :

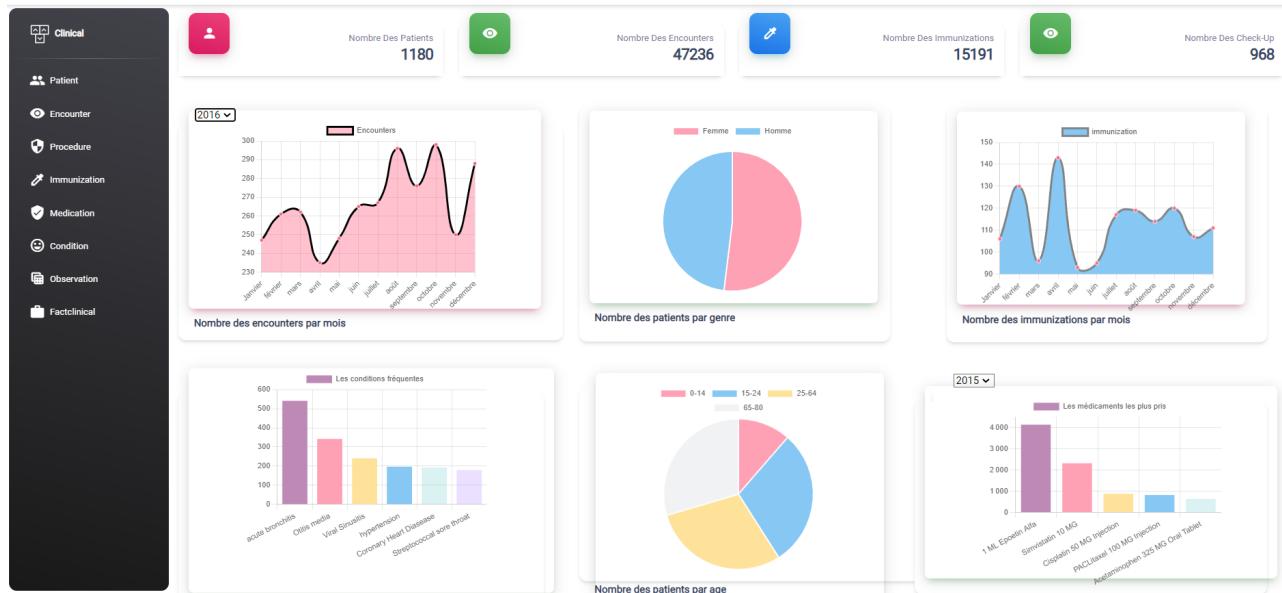


FIGURE 6.78 – Vue globale du tableau de bord

Le tableau de bord fournit une vue d'ensemble à travers cette représentation graphique. Le rapport fournit des statistiques clés et met en valeur des informations importantes sur l'état de santé des patients. Il montre le nombre total des patients, ainsi qu'une pie qui montre ce nombre par tranche d'âge et par sexe. De plus, il identifie les conditions les plus courantes pour lesquelles les patients consultent leur médecin, ainsi que les médicaments les plus prescrits. On trouve également un graphique de ligne montrant l'évolution du nombre des encounters par mois en sélectionnant une année. Cette visualisation permet une meilleure compréhension de la situation actuelle et aide à prendre des décisions éclairées.

6.5 Conclusion :

Dans ce chapitre, nous avons exposé les APIs qui assurent l'intégration facile des données, favorisant l'interopérabilité. De même, nous avons développé une application web avec un tableau de bord interactif qui fournit aux professionnels de santé une vision globale des données médicales, facilitant ainsi la prise de décision et l'amélioration des soins aux patients.

chapitre 7

Sprint 5 : Assistant de recommandation des médicaments

7.1 Introduction :

Dans ce chapitre, nous allons explorer la création d'un assistant de recommandation de médicaments pour les médecins en utilisant des techniques d'apprentissage automatique. Nous allons tout d'abord présenter l'ensemble de données que nous avons utilisé. Ensuite, nous allons présenter les techniques d'apprentissage automatique que nous avons utilisées et décrire le processus de formation de notre modèle de recommandation de médicaments. Enfin, nous allons présenter les résultats de notre modèle et également discuter des applications pratiques dans le domaine de la santé et de la médecine.

7.1.1 définition :

Le système de recommandation[22] en général, y compris le systèmes de recommandation de médicaments, est un algorithme d'intelligence artificielle qui utilise des mégadonnées pour recommander des éléments aux utilisateurs en fonction de leurs préférences et de l'historique de leurs activités. Les systèmes de recommandation sont très utiles car ils aident les utilisateurs à découvrir des produits et des services qu'ils ne trouveraient peut-être pas autrement. Cependant, il est important de noter que les systèmes de recommandation de médicaments comportent des responsabilités importantes et doivent être utilisés avec prudence par les professionnels de la santé pour garantir une prescription de médicaments sûre et efficace.

7.1.2 Les types des systèmes de recommandation :

- ♣ **Le filtrage collaboratif :** Les algorithmes de filtrage collaboratif recommandent des éléments en fonction des informations de préférence de nombreux utilisateurs. Cette approche utilise les préférences d'autres utilisateurs similaires pour recommander des éléments, l'algorithme de recommandation apprend à prédire les interactions futures. Ces systèmes de recommandation créent des modèles basés sur le

comportement passé d'un utilisateur, tels que des articles achetés précédemment ou des évaluations sur ces articles et des décisions similaires prises par d'autres utilisateurs.

- ♣ **Le filtrage de contenu :** Cet algorithme utilise les caractéristiques de l'élément lui-même pour recommander des éléments similaires aux préférences de l'utilisateur. Cette approche modélise de nouvelles probabilités basées sur la similarité des caractéristiques de l'article et de l'utilisateur, compte tenu des informations sur les utilisateurs et les articles avec lesquels ils interagissent .

7.2 Objectif de l'assistant de recommandation des médicaments :

L'objectif principal de l'assistant de recommandation de médicaments est d'aider les professionnels de la santé à prescrire l'ordonnance d'un patient concernant une tel maladie. À l'aide de techniques d'analyse de données et d'apprentissage automatique, l'assistant peut recommander des médicaments pour des maladies spécifiques sur la base de preuves scientifiques solides et de directives de traitement établies. En fait, l'assistant de recommandation de médicaments peut aider les médecins à choisir le médicament le plus approprié pour traiter une maladie spécifique en fonction des antécédents médicaux du patient, des allergies et des effets secondaires connus des médicaments. Cela optimise le traitement et réduit le risque d'effets secondaires indésirables.

En fait, l'assistant de recommandation de médicaments peut être utilisé pour aider les médecins à choisir le médicament le plus approprié pour traiter une maladie spécifique en fonction des antécédents médicaux du patient, des antécédents d'allergies et des effets secondaires connus des médicaments. Cela optimise le traitement et réduit le risque d'effets secondaires indésirables.

Cette figure présente le processus de recommandation des médicaments proposés :

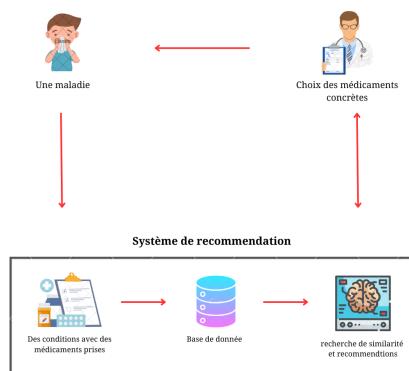


FIGURE 7.79 – Le processus de recommandation des médicaments

Il est important de mentionner que cet assistant de recommandation de médicaments n'est pas destiné à remplacer les prescriptions d'un professionnel de la santé, mais sert plutôt de guider. En fin de compte, l'objectif de l'assistant de recommandation de médicaments est d'améliorer la qualité des soins et favoriser le bien-être des patients.

7.3 Ensemble de données utilisé pour développer l'assistant :

7.3.1 Préparation des données :

L'ensemble de données utilisé pour développer l'assistant de recommandation de médicaments doit inclure des informations détaillées sur les traitements prescrits pour chaque maladie. Les données peuvent être collectées à partir de différentes sources, telles que les dossiers médicaux , les systèmes de pharmacie, les laboratoires... Dans notre cas, on va utiliser une dataset du domaine clinical qui provient déjà du standard HL7 FHIR et qui assure la relation entre les médicaments proposée par le médecin “medication” et les maladies en sortes de “condition”.

ID_Condition	ConditionText	PatientUID	Date	ID_Medication	MedicationText
0	Fracture of clavicle	5cbc121b-cd71-4428-b8b7-31e53eba8184	2016-05-21	1.0	Ibuprofen 200 MG Oral Tablet
1	Acute bronchitis (disorder)	adccf2c3-9dc4-4067-ba23-98982c4875da	2010-07-27	2.0	Acetaminophen 325 MG Oral Tablet
2	Acute bronchitis (disorder)	adccf2c3-9dc4-4067-ba23-98982c4875da	2012-01-26	2.0	Acetaminophen 325 MG Oral Tablet
3	Otitis media	31191928-6acb-4d73-931c-e601cc3a13fa	2017-10-24	5.0	Amoxicillin 500 MG Oral Tablet
4	Otitis media	31191928-6acb-4d73-931c-e601cc3a13fa	2017-10-24	1.0	Ibuprofen 200 MG Oral Tablet
...
3670	Atrial Fibrillation	d49615b1-38b4-4c37-b59c-076636983fb7	2001-12-24	94.0	Warfarin Sodium 5 MG Oral Tablet
3671	Atrial Fibrillation	d49615b1-38b4-4c37-b59c-076636983fb7	2001-12-24	95.0	Verapamil Hydrochloride 40 MG
3672	Atrial Fibrillation	d49615b1-38b4-4c37-b59c-076636983fb7	2001-12-24	96.0	Digoxin 0.125 MG Oral Tablet
3673	Streptococcal sore throat (disorder)	c588a992-d308-4916-aed5-389417e5f6e3	2009-10-26	28.0	Penicillin V Potassium 500 MG Oral Tablet
3674	Viral sinusitis (disorder)	c588a992-d308-4916-aed5-389417e5f6e3	2014-09-14	7.0	Amoxicillin 250 MG / Clavulanate 125 MG Oral T...

FIGURE 7.80 – les données utilisées pour développer l'assistant de recommandation des médicaments

7.3.2 Exploration des données :

L'exploration de données est une étape importante pour comprendre les données avec lesquelles les gens travaillent et identifier les tendances, les relations et les modèles qui peuvent ne pas être évidents à première vue, ce qui peut améliorer la qualité de l'analyse et les informations obtenues. On a utilisé les bibliothèques plotly.express et matplotlib pour afficher certains détails .

Dans ce cadre, on va disposer quelques statistiques que nous avons fait afin de découvrir les données dominantes dans notre base de donnée .

Voici un diagramme circulaire qui présente les 10 conditions de prise de médicaments les plus fréquentes :

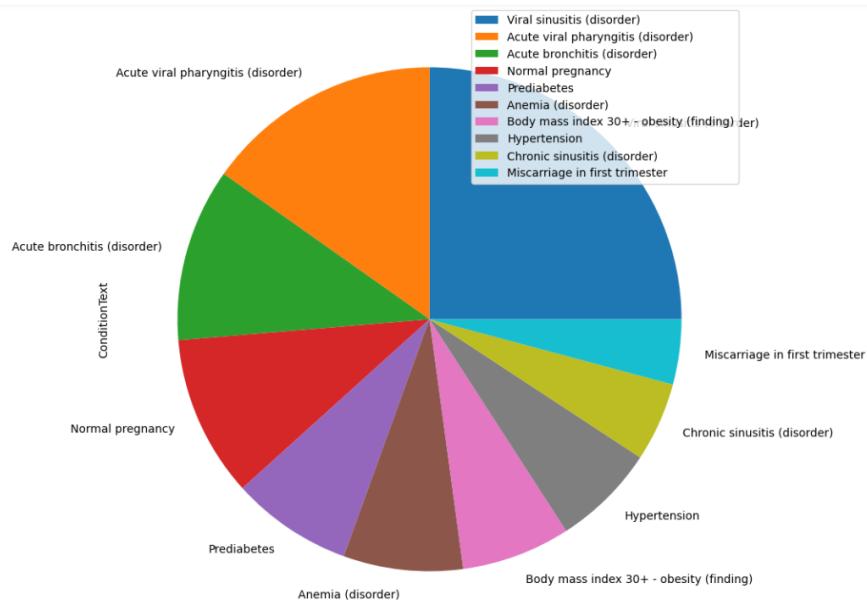


FIGURE 7.81 – les 10 conditions de prise de médicaments les plus fréquentes

On présente également l'histogramme qui visualise les 20 médicaments fréquemment pris :

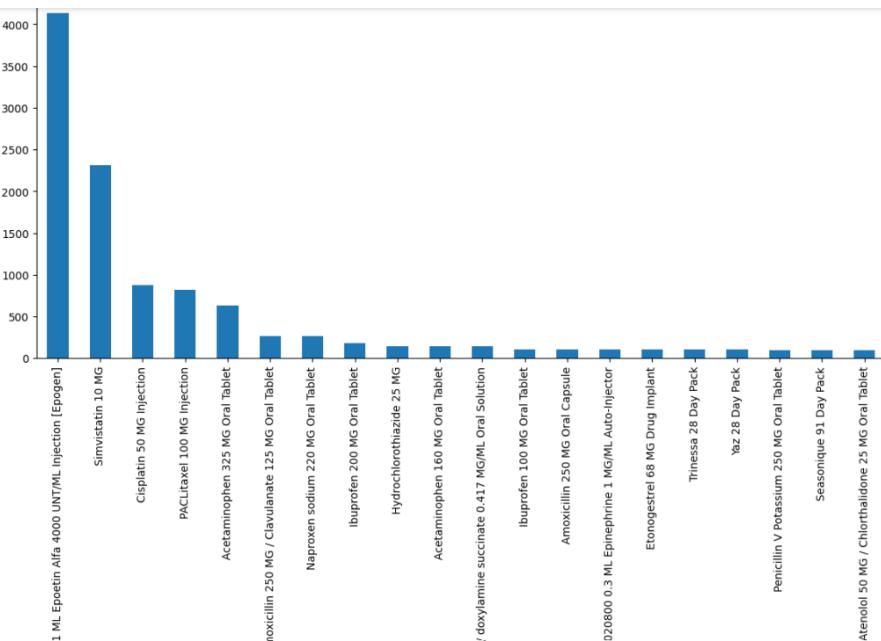


FIGURE 7.82 – les 20 médicaments les plus pris

Finalement ,on a utilisé la bibliothèque **Plotly Express** pour créer des graphiques interactifs riches avec des animations et des tendances. Dans ce sens , nous avons crée une carte graphique exposant les conditions de la prise des médicaments relativement aux médicaments pris.

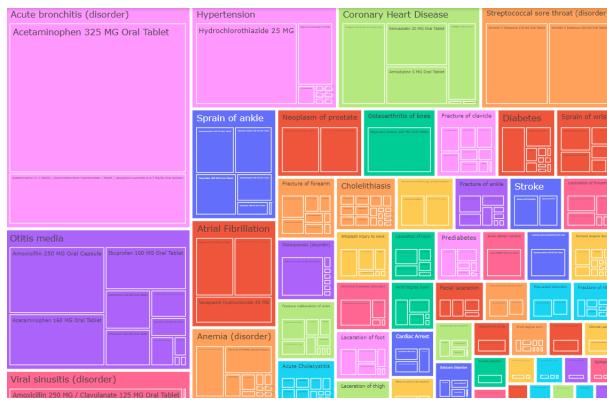


FIGURE 7.83 – les médicaments recommandés relatifs aux maladies

Les données hiérarchiques sont souvent stockées sous forme de tableaux rectangulaires, avec des colonnes différentes correspondant à différents niveaux de la hiérarchie. Pour notre figure , chaque rectangle correspond à une condition et qui porte des niveaux de hiérarchie selon le nombre des médicaments recommandés par le médecin lors de chaque condition .

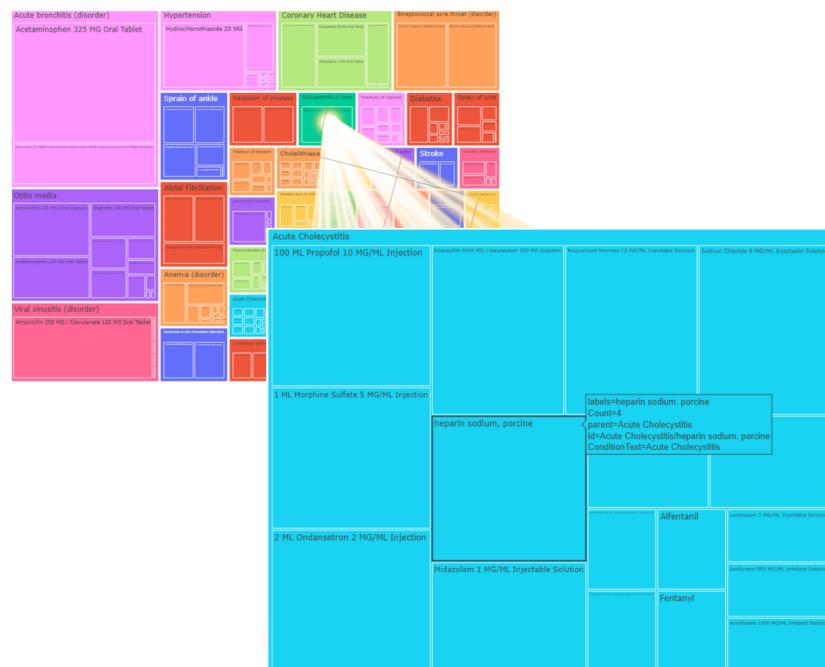


FIGURE 7.84 – Un exemple des médicaments recommandés relatif à une maladie

7.4 Comportement des données :

Le comportement des données est un domaine crucial de l'analyse de données qui consiste à comprendre les modèles et les relations au sein des données. Dans le cadre de l'analyse médicale, le partitionnement (Clustering) est une méthode clé pour identifier des groupes de médicaments qui sont souvent prescrits ensemble pour traiter des maladies similaires.

En utilisant des techniques de partitionnement des données (clustering) et de matrice de factorisation, il est possible d'améliorer la précision et la pertinence des recommandations de médicaments en permettant une classification plus fine et plus précise des médicaments. Cette approche peut aider les professionnels de la santé à choisir les traitements les plus efficaces pour leurs patients et à mieux comprendre les interactions médicamenteuses potentielles, améliorant ainsi la qualité des soins médicaux.

7.4.1 Classification :

Pour utiliser l'algorithme K-Means, on doit d'abord spécifier le nombre de partitions (clusters) K et puis l'implémenter dans notre approche afin de découvrir les relations des conditions

♣ La recherche du nombre de clusters :

Cependant, la méthode Elbow est l'une des méthodes les plus populaires. Elle consiste à exécuter K-Means pour une gamme de valeurs de K groupes et à calculer l'inertie intra-cluster pour chaque groupe. Lorsque les distorsions sont tracées et que la courbe ressemble à un bras, le coude (point d'infexion de la courbe) est la meilleure valeur de K à partir duquel la variance ne baisse plus significativement. Nous avons alors obtenu cette figure qui montre que le nombre de partition(clusters) optimal est égal à 2.

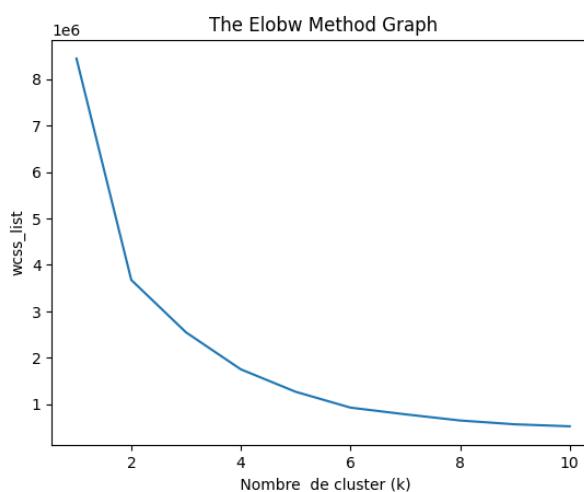


FIGURE 7.85 – Méthode graphique de Elbow

♣ Application de l'algorithme K-Means :

Ce graphe consiste à mettre en œuvre l'algorithme de K-Means et la disposition de nos données :

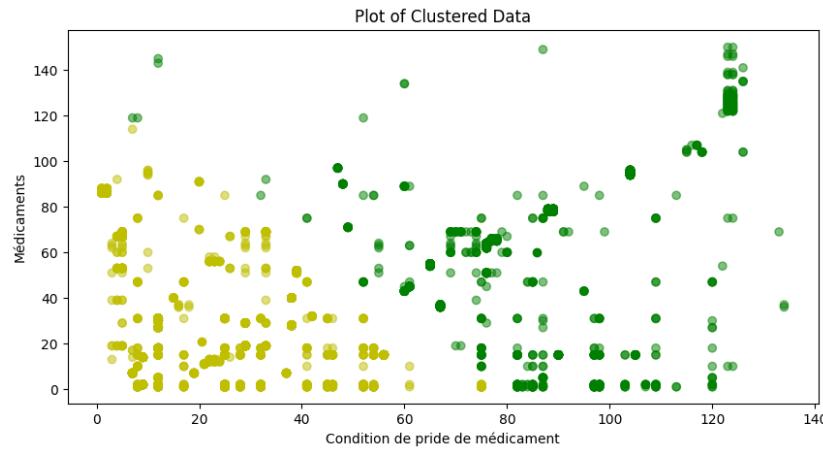


FIGURE 7.86 – Visualisation des clusters

7.4.2 Matrice de factorisation :

Cet algorithme est une implémentation de la matrice de factorisation en utilisant la bibliothèque PyTorch pour créer un système de recommandation de médicaments pour les maladies.

On commence par créer la classe `MatrixFactorization`. C'est une sous-classe de `torch.nn.Module` qui contient les paramètres et les méthodes nécessaires pour entraîner et utiliser le modèle. Le modèle utilise deux tableaux d'embedding, un pour les maladies et l'autre pour les médicaments, qui sont multipliés ensemble pour produire les scores de recommandation. La fonction `predict` permet de faire des prédictions à partir des identifiants "ID" des maladies et des médicaments.

```
import torch
import numpy as np
from torch.autograd import Variable
from tqdm import tqdm_notebook as tqdm

class MatrixFactorization(torch.nn.Module):
    def __init__(self, n_desease, n_medicaments, n_factors=20):
        super().__init__()
        # create user embeddings
        self.desease_factors = torch.nn.Embedding(n_desease, n_factors) # think of this as a
        # create item embeddings
        self.medicament_factors = torch.nn.Embedding(n_medicaments, n_factors) # think of th
        self.desease_factors.weight.data.uniform_(0, 0.05)
        self.medicament_factors.weight.data.uniform_(0, 0.05)

    def forward(self, data):
        # matrix multiplication
        desease, medicaments = data[:,0], data[:,1]
        return (self.desease_factors(desease)*self.medicament_factors(medicaments)).sum(1)
        # def forward(self, user, item):
        #     # matrix multiplication
        #     return (self.user_factors(user)*self.item_factors(item)).sum(1)

    def predict(self, desease, medicaments):
        return self.forward(desease, medicaments)
```

FIGURE 7.87 – Code de la classe `MatrixFactorisation`

♣ Optimisation :

On poursuit de créer un code qui initialise le modèle de factorisation matricielle et configure son entraînement. Il définit également les hyperparamètres, tels que le nombre d'époques, la disponibilité de GPU, la fonction de perte et l'optimiseur pour les données d'entraînement.

Finalement, nous avons arrivé à obtenir cette courbe de perte des données qui poursuit à diminuer et tend vers 0. Cela confirme que notre modèle est bien entraîné.

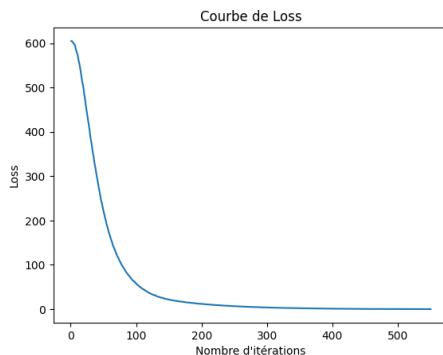


FIGURE 7.88 – Courbe de perte des données

Notez bien : La complexité croissante de ces modèles nécessite des ressources informatiques importantes, notamment des GPU. Ces unités de traitement graphique sont essentielles pour accélérer les calculs et gérer les charges de travail intensives.

Finalement, nous avons effectué un regroupement (clustering) des médicaments en utilisant l'algorithme K-Means. Ensuite, pour chaque groupe, le code récupère les noms des médicaments qui ont été assignés à ce cluster.

La figure suivante présente des médicaments chacun assignés à son groupe :

```

Cluster #0
Ibuprofen 100 MG Oral Tablet
Trinessa 28 Day Pack
Hydrochlorothiazide 25 MG
Seasonique 91 Day Pack
Yaz 28 Day Pack
Ibuprofen 200 MG Oral Tablet
Acetaminophen 325 MG Oral Tablet
10 ML oxaliplatin 5 MG/ML Injection

Cluster #1
Acetaminophen 325 MG Oral Tablet
Simvistatin 10 MG
Ibuprofen 200 MG Oral Tablet
Simvistatin 10 MG
Simvistatin 10 MG
  
```

FIGURE 7.89 – Résultat de regroupement des médicaments

7.5 Techniques d'apprentissage automatique pour la recommandation des médicaments :

La recommandation de médicaments basée sur l'historique médical des patients et les médicaments recommandés par les médecins dans différentes conditions est un enjeu complexe qui peut être résolu à l'aide d'algorithmes de recommandation. Pour cela, deux approches sont souvent utilisées : la vectorisation et la matrice d'interaction. Nous présentons alors l'implémentation de ces deux approches en utilisant la bibliothèque scikit-learn de Python.

7.5.1 Réalisation du premier algorithme :

La première approche implique l'utilisation de techniques de traitement du langage naturel pour représenter des éléments sous forme de vecteurs de mots. Ensuite, on utilise la mesure de similarité pour trouver les éléments les plus similaires. Dans notre contexte, on veut transformer les textes des conditions en vecteurs numériques normalisés, puis mesurer leur similitude en fonction des médicaments pris dans ces contextes.

Recherche de similarité :

La fonction **CountVectorizer** permet de convertir une collection des expressions en vecteur en fonction de nombre de répétitions des mots. Après avoir converti les expressions en vecteurs, on veut savoir si ces deux vecteurs sont similaires ou pas grâce à la distance entre eux. Par distance, nous entendons la distance angulaire entre deux vecteurs, représentée par θ (thêta). En réfléchissant davantage du point de vue de l'apprentissage automatique, nous comprenons que la valeur de $\cos \theta$ a plus de sens pour nous que la valeur de θ car la fonction cosinus (ou "cos") va mapper la valeur de θ dans le premier quadrant entre 0 et 1.

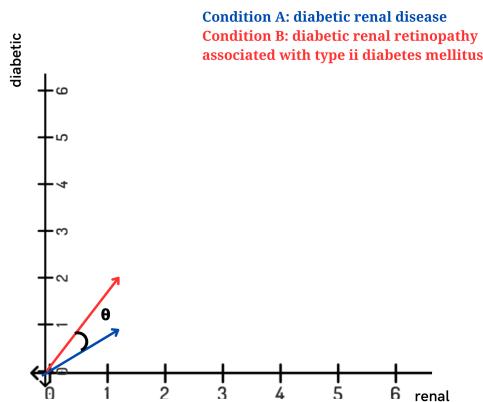


FIGURE 7.90 – Présentation d'un exemple de vecteurs

Afin de trouver un moyen de représenter nos conditions sous forme de vecteurs, on a fait recours à la classe **CountVectorizer()** de la bibliothèque `sklearn.feature_extraction.text`. Par la suite, on appelle la fonction `cosine_similarity()` de la bibliothèque `sklearn.metrics.pairwise` pour calculer la similarité.

Recommandation :

Nous avons créé une fonction qui permet de récupérer le médicament, trouver l'index dans la matrice de similarité et énumérer les scores de similarité de toutes les autres conditions par rapport à celui-ci.

7.5.2 Réalisation du deuxième algorithme :

La deuxième approche d'aide à la suggestion de médicaments utilise de même la similarité cosinus pour recommander des médicaments, mais elle celle-ci est fondée sur les données de la matrice d'interaction .Cette matrice fonctionne en créant un tableau croisé (crosstable) de l'ensemble des conditions et qui indique le nombre de fois où chaque médicament a été pris en fonction des maladies. Cette matrice est ensuite transformée en une matrice de similarité en utilisant la similarité cosinus (`cosine_similarity`) pour mesurer la similitude entre chaque paire de conditions(maladies).Les conditions similaires auront des scores de similarité plus élevés, tandis que les conditions moins similaires auront des scores moins élevés .

Ensuite, les médicaments les plus similaires sont sélectionnés en fonction d'un seuil de similarité prédéfini. Cette approche permet de recommander des médicaments qui sont similaires en termes d'interactions avec les autres éléments de la matrice d'interaction. Cela peut aider à recommander des médicaments plus adaptés à chaque cas spécifique, en fonction des interactions connues entre les médicaments et les autres entités de la matrice.

Recommandation :

Ce code permet de recommander des médicaments en fonction d'une maladie d'entrée. Il récupère les rencontres similaires pour cette maladie d'entrée, puis pour chaque rencontre similaire, il extrait les médicaments prescrits associés à cette rencontre. Enfin, il retourne une liste de tous les médicaments prescrits pour les rencontres similaires récupérées.

7.5.3 Tester les 2 approches :

Afin de tester les 2 approches , on va entrer une maladie et découvrir les médicaments suggérées s'ils sont convergents. La première figure illustrera la maladie entré :

```
input_disease = 'fracture of clavicle'
```

FIGURE 7.91 – Un exemple de maladie à tester

La deuxième figure illustre les médicaments recommandé par la première approche :

```
recommendations_2 = Vec_recommend_medications(input_disease, num_recommendations=5)
print(set(recommendations_2))

{'acetaminophen 160 mg oral tablet', 'ibuprofen 200 mg oral tablet', 'meperidine hydrochloride 50 mg oral tablet'}
```

FIGURE 7.92 – Exemple de recommandation avec le premier algorithme

La troisième figure illustre les médicaments recommandé par la deuxième approche :

```
recommendations_1 = inter_recommend_medications(input_disease)
print(set(recommendations_1))

{'acetaminophen 160 mg oral tablet', 'acetaminophen 325 mg oral tablet', 'ibuprofen 200 mg oral tablet', 'naproxen sodium 220 mg oral tablet', 'ibuprofen 100 mg oral tablet'}
```

FIGURE 7.93 – Exemple de recommandation avec le deuxième algorithme

7.6 Conclusion :

En conclusion, ce chapitre a examiné plusieurs approches pour recommander des médicaments. L’analyse a montré que ces approches peuvent fournir des recommandations utiles pour le traitement d’une condition médicale cible, mais elles présentent également certaines limites qui doivent être prises en compte. Les limites incluent les caractéristiques des patients , les critères d’évaluation utilisés et les traitements alternatifs. Il est important de tenir compte de ces limites lors de l’interprétation et de l’application des résultats d’analyse.

Conclusion Générale

Ce travail a été le fruit d'un effort considérable et d'un engagement soutenu pendant quatre mois au sein de l'entreprise ITProgress. Nous avons investi notre temps et nos compétences pour mener à bien ce projet complexe et ramifié.

La construction d'un data mesh peut être un processus complexe, mais c'est un élément crucial pour la transformation numérique réussie dans le domaine de la santé. Notre projet a démontré comment la mise en place d'un data mesh peut aider à optimiser l'utilisation des données en créant des domaines centrés sur les besoins métier spécifiques et inspirés du niveau 3 et 4 du modèle standards HL7 FHIR . De ce fait , nous avons identifié trois domaines : Clinical, Individuals et Financial, chacun avec des besoins et des cas d'utilisation distincts.

Le département Individuals a créé une base de données orientée graphe implémenté dans Neo4j , qui permet d'identifier les relations entre les patients, les médecins et leurs localisations. Cette base de données permettra une meilleure compréhension des flux de patients et facilitera la planification de la prise en charge des patients.

Le département Financial a créé un entrepôt de données dans PostgreSQL .Il tendra à suivant de travailler sur un algorithme qui permettra de prédire si un patient est éligible à un remboursement.

Le département Clinical, quant à lui, a créé un entrepôt de données et chargé ses données dans PostgreSQL. On a travaillé sur la création d'un tableau de bord et d'un assistant de recommandation des médicaments. L'assistant de recommandation des médicaments utilise des données pour recommander des traitements spécifiques aux maladies. Une des perspectives futures intéressantes serait de considérer l'analyse des données de l'ensemble du dossier médical électronique du patient, y compris les antécédents médicaux, les allergies, les résultats de tests de laboratoire, les notes de consultation, les médicaments précédemment prescrits, et les évaluations de l'état de santé global. Cette approche pourrait permettre une personnalisation encore plus grande de la recommandation de médicaments, en prenant en compte un plus grand nombre de variables et de caractéristiques individuelles du patient.D'une autre part , on tendrait d'implémenter l'algorithme de l'assistant de recommandation des médicament dans notre application web.

Conclusion Générale

Enfin, pour continuer à améliorer notre data mesh, il sera important de construire une plateforme robuste qui facilite l'accès et la gestion des données dans tous les domaines. Nous devrons également travailler à renforcer la gouvernance des données et la sécurité, tout en favorisant la collaboration entre les différents domaines et les équipes impliquées.

En résumé, ce projet nous a permis de travailler sur différents aspects de la gestion des données en santé, en mettant en place des solutions spécifiques à chaque domaine. Nous sommes convaincus que ces solutions permettront d'améliorer la qualité des soins, tout en facilitant le travail des professionnels de santé.

Table des annexes

Annexe 1 : Les bibliothèques Python utilisées

Annexe 2 : HL7 FHIR

Annexe 3 : Synthéa

Les bibliothèques Python utilisées

Fhir.Resources

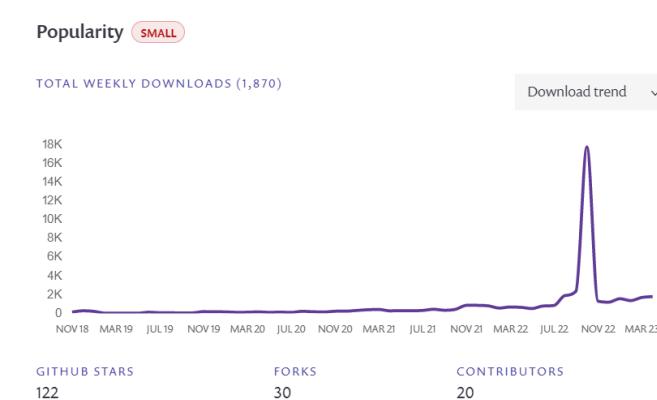


FIGURE 94 – Croissance du téléchargement de la bibliothèque Fhir resources

Le package "fhir.resources" est un projet open source qui fournit des outils et des classes pour manipuler les ressources FHIR en Python. Il est distribué sous licence BSD et a été initialement publié en 2020.

Le package "fhir.resources" a été créé par Houda El Fadil, un développeur de logiciels qui travaille sur des projets de santé numérique. Les contributeurs ont ajouté des fonctionnalités telles que la prise en charge de la sérialisation et de la désérialisation en JSON et XML, ainsi que l'ajout de classes de ressources FHIR plus récentes.

En plus des contributeurs, le package "fhir.resources" s'appuie sur les technologies open source pydantic et orjson pour fournir une validation de données rapide et efficace. Pydantic est une bibliothèque de validation de données qui permet de créer des classes de données Python à partir de modèles de données, tandis qu'orjson est une bibliothèque de sérialisation et de désérialisation JSON rapide.

Cas pratique : Il suffit d'ouvrir un fichier json en python après importer les librairies. L'objet JSON est alors converti en un objet Bundle de la bibliothèque Python FHIR. Une fois que l'objet Bundle a été créé, les ressources sont extraites et stockées dans une liste appelée resources. Une boucle for est utilisée pour parcourir chaque ressource et stocker son type dans une autre liste appelée oneResources.

```

f = open('E:\\pfe\\k_fhir\\' + filesList[40] ,)
json_obj = json.load(f)
oneBundle = Bundle.parse_obj(json_obj)
# Resources
resources = []
if oneBundle.entry is not None:
    for entry in oneBundle.entry:
        resources.append(entry.resource)

oneResources = []
for j in range(len(resources)):
    oneResources.append(type(resources[j]))

print(len(oneResources))

uniqResources = set(oneResources)
print(len(uniqResources))
uniqResources

```

273
15

{fhir.resources.careplan.CarePlan,
fhir.resources.careteam.CareTeam,
fhir.resources.claim.Claim,
fhir.resources.condition.Condition,
fhir.resources.diagnosticreport.DiagnosticReport,
fhir.resources.encounter.Encounter,
fhir.resources.explanationofbenefit.ExplanationOfBenefit,
fhir.resources.goal.Goal,
fhir.resources.immunization.Immunization,
fhir.resources.medicationrequest.MedicationRequest,
fhir.resources.observation.Observation,
fhir.resources.organization.Organization,
fhir.resources.patient.Patient,
fhir.resources.practitioner.Practitioner,
fhir.resources.procedure.Procedure}

FIGURE 95 – Exemple de code utilisant fhir.resources

Merge() :

La fonction merge() de Pandas DataFrame est utilisée pour fusionner deux objets DataFrame en utilisant une opération de jointure de style base de données. La jointure est effectuée sur les colonnes ou les index. Si la jointure est effectuée sur les colonnes, les index sont ignorés. Cette fonction renvoie un nouveau DataFrame, et les objets DataFrame source ne sont pas modifiés.

Syntaxe :

```
pd.merge(left, right, how='inner', on=None, left_on=None,right_on=None,
left_index=False,right_index=False, sort=True)
```

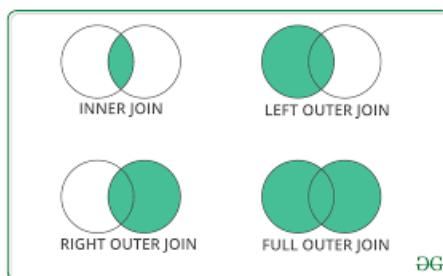
Type de Merge() :

FIGURE 96 – Type de jointure en python

HL7 FHIR



FIGURE 97 – Logo de HL7 fhir

Site officiel : <http://hl7.org/fhir/>

HL7 FHIR [B3] est un standard de santé électronique qui permet l'interopérabilité des systèmes informatiques de santé. Il a été développé par Health Level Seven International (HL7), une organisation internationale à but non lucratif qui se consacre à la normalisation des technologies de l'information dans le domaine de la santé.

Le standard FHIR utilise une approche basée sur les ressources pour représenter les données de santé. Une ressource est une unité d'information qui décrit un élément spécifique de l'ensemble de données de santé, tel qu'un patient, un praticien, un médicament ou une observation. Chaque ressource est définie de manière standardisée, ce qui facilite l'échange d'informations entre les différents systèmes de santé.

FHIR utilise également une architecture RESTful pour faciliter l'échange de données entre les systèmes. REST signifie Representational State Transfer et c'est une architecture de communication qui permet aux différents systèmes de communiquer entre eux à l'aide de messages HTTP standard. Les API RESTful sont simples à utiliser et bien documentées, ce qui facilite l'adoption du standard FHIR par les professionnels de la santé.

Un autre avantage de FHIR est sa flexibilité et son évolutivité. Le standard est conçu pour être facilement extensible, ce qui permet aux organisations de santé de l'adapter à leurs besoins spécifiques. Par exemple, une organisation de santé peut ajouter des champs de données supplémentaires à une ressource FHIR existante pour répondre à ses besoins spécifiques.

En résumé, HL7 FHIR est un standard de santé électronique important qui facilite l'interopérabilité des systèmes informatiques de santé. Il utilise une approche basée sur les ressources et une architecture RESTful pour permettre l'échange de données entre les différents systèmes. FHIR est également évolutif, ce qui permet aux organisations de santé de l'adapter à leurs besoins spécifiques.

Philosophie de Grahame pour HL7 FHIR [B3]

Le développement de la norme FHIR est guidé par la vision de Grahame et les décisions sont prises par consensus. La philosophie de Grahame repose sur un modèle de données simplifié pour les soins de santé, composé de ressources dont le contenu est intentionnellement limité. La règle des 80% de FHIR stipule qu'une ressource ne doit contenir que les éléments de données acceptés par 80% ou plus des participants. L'approche de FHIR privilégie l'utilisabilité par rapport à la complétude. Le développement de la norme est guidé par les développeurs de logiciels qui ont travaillé sur des problèmes réels, et la vérification de la norme se fait par l'utilisation de technologies web intersectorielles. Cette approche rend la norme plus accessible et attractive pour les développeurs modernes et permet la création d'applications mobiles et web basées sur FHIR.

Ressources de HL7 fhir

Le standard FHIR est composé de différentes ressources, qui peuvent être de niveaux de maturité différents, allant de 0 (projet expérimental) à 5 (normatif). Les ressources normatives sont considérées comme étant des standards éprouvés et sont les plus importantes. Le nombre de ressources FHIR devrait augmenter avec le temps et les ressources existantes peuvent être modifiées. La spécification FHIR utilise un modèle de données organisé en ressources modulaires. La version 4.01 actuelle de la norme répartit les 145 ressources en cinq niveaux (Foundation, Base, Clinical, Financial, Specialized) avec des sous-catégories pour chacun.

- **Le niveau 1 :** est le cadre de base (30 ressources) avec des sous-catégories telles que Conformité, Terminologie, Sécurité, Documents, etc.
- **Le niveau 2 :** (26 ressources) fournit un support pour la mise en œuvre et la liaison avec des spécifications externes.
- **Le niveau 3 :** (39 ressources) couvre les éléments structurels et de processus des systèmes de soins de santé réels.
- **Le niveau 4 :** (16 ressources) concerne la tenue de registres et l'échange de données.
- **Le niveau 5 :** (35 ressources) permet de raisonner sur les processus de soins de santé. Bien que cette organisation soit utile pour comprendre la portée de la norme, elle ne couvre pas toutes les ressources existantes ou futures.

La figure représentée englobe les modules et leurs exemples de ressources



FIGURE 98 – Les ressources en HL7 fhir[3]

Synthéa



FIGURE 99 – Logo de Synthéa[4]

Site Officiel : sythaea.fr

Synthea[B4] est un framework puissant qui utilise une approche modulaire pour générer différents aspects des données de santé. Cela permet aux ingénieurs et scientifiques des données d'adapter le système en fonction des populations de patients locales si nécessaire. Leur documentation fournit de bons exemples et guides pour créer vos propres modules.

Synthea est un simulateur/générateur de données open source qui génère des données de santé réalistes. Il propose également des ensembles de données pré-générées téléchargeables depuis son site web. Ces ensembles de données sont entièrement synthétiques, ne contenant aucune donnée réelle sur les patients. Cela facilite le travail avec les données, notamment lors de l'inclusion de différents formats ou modèles de données dans les tests unitaires et d'intégration.

En outre, le site web de Synthèa propose une communauté en ligne pour les utilisateurs de Synthèa, où ils peuvent discuter de leurs expériences et partager leurs résultats avec d'autres utilisateurs de Synthèa. Il y a également des forums de discussion, des ressources de formation et des projets collaboratifs pour les personnes intéressées par l'utilisation de Synthèa. Le site web de Synthèa est une ressource précieuse pour ceux qui cherchent à utiliser des données de santé synthétiques dans leurs projets de recherche et de développement.

Étant donné que d'ici 2024, 60% des données utilisées pour développer des solutions d'IA et d'analyse seront synthétiquement générées, nous avons opté pour l'utilisation de données Synthea pour notre projet de recherche en santé.[5]

Nous avons téléchargé des données de Synthea pour notre projet de recherche en santé. Nous avons choisi d'utiliser les données de Synthea car elles sont libres de coût, de restrictions de confidentialité et de sécurité, et elles représentent une population entière avec une grande variété de caractéristiques démographiques et de conditions médicales. En utilisant les données de Synthea, nous espérons pouvoir fournir des informations utiles pour la prise de décisions en matière de santé et contribuer à l'amélioration des soins de santé pour tous.

Webographie

- [1] **Exemple de la réalisation d'une User Story** , [consulté le 01/03/2023] :
<https://agiliste.fr/les-specifications-agiles/t-1637062178772>.
- [2] **Méthode Scrum** , [consulté le 04/03/2023] : <https://agiliste.fr/guide-de-demarrage-scrum/>.
- [3] **Capture des ressources HL7 FHIR** , [consulté le 06/05/2023] :
<http://hl7.org/fhir/>.
- [4] **Logo de synthéa** , [consulté le 03/05/2023] :
<https://synthetichealth.github.io/synthea/about-landing>.
- [5] **Statistiques de Gartner** , [consulté le 03/03/2023] :
https://blogs.gartner.com/andrew_white/2021/01/12/our-top-data-and-analytics-predicts-for-2021/.
- [6] **Statistiques de Forrester**, [consulté le 03/03/2023] :
<https://www.forrester.com/blogs/hadoop-is-datas-darling-for-a-reason/>.
- [7] **Statistiques de Accenture** , [consulté le 04/03/2023] :
https://www.accenture.com/_acnmedia/pdf-108/accenture-closing-data-value-gap-fixed.pdf.
- [8] **Méthode Classique**, [consulté le 28/02/2023] :
<https://www.appvizer.fr/magazine/operations/gestion-de-projet/methode-classique-gestion-de-projet>.
- [9] **Data Mesh** : <https://www.datamesh-architecture.com/> , [consulté le 20/02/2023].
- [10] **Google Collaboratory** , [consulté le 02/05/2023] : <https://research.google.com/colaboratory/faq.html>.
- [11] **IntelliJ IDEA**, [consulté le 15/05/2023] : <https://www.jetbrains.com/help/idea/discover-intellij-idea.htmlmulti-platform-IDE>.
- [12] **Google Drive**, [consulté le 20/05/2023] : <https://www.techtarget.com/searchmobilecomputing/definition/Google-Drive>.
- [13] **Draw.io** , [consulté le 20/05/2023] : <https://www.tice-education.fr/tous-les-articles-er-ressources/articles-internet/819-draw-io-un-outil-pour-dessiner-des-diagrammes-en-ligne>.
- [14] **Neo4j** , [consulté le 17/03/2023] : <https://neo4j.com/fr/>.
- [15] **OrientDB** , [consulté le 17/03/2023] : <http://orientdb.org/docs/3.0.x/misc/Overview.html>.
- [16] **OrientDB vs Neo4j** , [consulté le 13/03/2023] : <https://db-engines.com/en/system/Neo4j%3BOrientDB>.

- [17] **Partitionnement PostgresQL** , [consulté le 13/03/2023] : <https://docs.postgresql.fr/10/ddl-partitioning.html>.
- [18] **Jhipster** , [consulté le 21/03/2023] : <https://www.processindustries.fr/tout-savoir-sur-jhipster/>.
- [19] **JDL** , [consulté le 22/03/2023] : <https://www.jhipster.tech/jdl/intro>.
- [20] **Angular** <https://angular.io/guide/what-is-angular> , [consulté le 11/05/2023].
- [21] **Spring Boot** , [consulté le 01/05/2023] : <https://stackify.com/what-is-spring-boot/>.
- [22] **Système de recommandation** , [consulté le 05/05/2023] : https://www-nvidia-com.translate.goog/en-us/glossary/data-science/recommendation-system/?_x_tr_sl=en_x_tr_tl=fr_x_tr_hl=fr_x_tr_pto=rqfbclid=IwAR0izh1awvqaGQyxTftnZc7WUaDfvVK1Lh5EaU2P7hnG5jOxvvpMu4hY27o:text=A%20recommendation%20system%20.

Bibliographie

[B1] **Data Mesh : Delivering Data- Driven Value at Scale** écrit par Zhamak Dehghani.[en ligne] ,[consulté le 01/03/2023], via l'adresse : <https://www.manning.com/freebook>

[B2] **Data Mesh in Action** écrit par Jacek Majchrzak, Sven Balnojan, and Marian Siwiak with Mariusz Sieraczkiewicz . [en ligne] ,[consulté le 01/03/2023], via l'adresse : <https://www.manning.com/freebook>

[B3] **Health Informatics on FHIR : How HL7's API is Transforming Healthcare** pour Mark L. Braunstein , [en ligne] ,[consulté le 07/02/2023], via l'adresse : <https://link.springer.com/book/10.1007/978-3-030-91563-6>

[B4] **Hands-On Healthcare Data Taming the Complexity of Real-World Data** pour Andrew Nguyen , première édition en Août 2022 [consulté en ligne via une plateforme] : <https://www.oreilly.com/cs/catalog/create/errata/?b=71910>.

[B5] **Data Mesh : Delivering Data-Driven Value at Scale** écrit par Zhamak Dehghani , [consulté en ligne le 01/03/2023 via une plateforme] : <https://www.oreilly.com/library/view/data-mesh/9781492092384/>.