



HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Département des Technologies de l'information et de la
communication (TIC)

Filière Télécommunications
Orientation Réseaux et Services

Travail de Bachelor

Développement d'une base de données des paramètres cliniques pour révéler des marqueurs et prise en charge adéquate de la douleur chirurgicale chez l'enfant

Étudiante

Francine Vanessa YOUNDZO KENGNE

Enseignant responsable

Prof.Dr. Silvia SCHINTKE, HEIG-VD

Entreprise mandante

Dr. Nathalya KOROGOD
HESAV-Haute Ecole de Santé Vaud
Avenue de Beaumont 21, 1011 Lausanne

Année académique

2019-2020

Yverdon-les-Bains, le 31 juillet 2020

Département des Technologies de l'information et de la communication (TIC)

Filière Télécommunications

Orientation Réseaux et Services

Étudiante : Francine Vanessa YOUNDZO KENGNE

Enseignant responsable : Prof.Dr. Silvia SCHINTKE, HEIG-VD

Travail de Bachelor 2019-2020

Développement d'une base de données des paramètres cliniques pour révéler des marqueurs et prise en charge adéquate de la douleur chirurgicale chez l'enfant

Nom de l'entreprise/institution

HESAV-Haute Ecole de Santé Vaud

Résumé publiable

Contexte et objectifs

Ce projet, issu d'une collaboration entre la HEIG-VD et la HESAV, a pour but le développement d'un prototype de base de données pour le milieu médical. Elle servira à améliorer l'évaluation et la prise en charge de la douleur en milieu hospitalier, ceci en particulier pour des nouveau-né-e-s et nourrissons subissant des interventions chirurgicales et qui ne peuvent pas encore explicitement exprimer leurs douleurs.

Développement de la base de données

Une base de données en architecture relationnelle a été développée se basant sur des outils libres de droits. Des routines d'importation des données existantes ont été programmées ainsi que des routines pour des saisies manuelles des données médicales issues de nouveaux examens médicaux. La gestion des droits d'accès sécurisés pour différentes catégories de personnel médical a été implémentée pour la saisie et la lecture.

Implémentation et tests

La base de données peut servir pour extraire des données d'un ou plusieurs patients pour le personnel médical, ou pour lister les examens conduits en vue de l'établissement de rapports de gestion. Des tests du backend sont concluants et une interface d'utilisateur est en développement afin de pouvoir conduire des tests avec des utilisateurs.

Conclusion et perspectives

La structure du prototype de la base de données développée dans le cadre du projet sera techniquement compatible avec le concept de Common Data Format du domaine médical et ainsi avec des réseaux existants, tels que SwissNeoNet ou PEDSnet. Ainsi la base de données pourra s'intégrer par la suite avec ces réseaux existants.

Étudiante :	Date et lieu :	Signature :
Francine Vanessa YOUNDZO KENGNE
Enseignant responsable :	Date et lieu :	Signature :
Prof.Dr. Silvia SCHINTKE, HEIG-VD
Nom de l'entreprise/institution :	Date et lieu :	Signature :
Dr. Nathalya KOROGOD
HESAV-Haute Ecole de Santé Vaud		

Préambule

Ce travail de Bachelor (ci-après TB) est réalisé en fin de cursus d'études, en vue de l'obtention du titre de Bachelor of Science HES-SO en Ingénierie.

En tant que travail académique, son contenu, sans préjuger de sa valeur, n'engage ni la responsabilité de l'auteur, ni celles du jury du travail de Bachelor et de l'Ecole.

Toute utilisation, même partielle, de ce TB doit être faite dans le respect du droit d'auteur.

HEIG-VD

Vincent Peiris
Chef de département TIC

Yverdon-les-Bains, le 31 juillet 2020

Authentication

La soussignée, Francine Vanessa YOUNDZO KENGNE, atteste par la présente avoir réalisé ce travail et n'avoir utilisé aucune autre source que celles expressément mentionnées.

Yverdon-les-bains, le 31 juillet 2020

Francine Vanessa YOUNDZO KENGNE

Dédicaces

*À mes parents **M Raphaël KENGNE** et **Mme Honorine Noël MAKAMTE** qui m'aiment, me soutiennent et se sacrifient afin que je devienne la personne que je suis aujourd'hui.*

*À mon fiancé **M Roderic ITAMBE**.*

*À mes frères et soeurs **Bekam, Beku, Pobet, Tcheutchoua et Tene**.*

*À ma tante **Mme Pascaline SCHLACHTER MAKAMTE**.*

Remerciements

Ce travail de Bachelor représente une étape majeure de ma vie, la récompense de plusieurs années de dur labeur. Rendu au terme de ces années de Bachelor, mes plus sincères remerciements et ma gratitude vont tout particulièrement :

A **Dieu** qui me donne chaque jour le souffle de vie et la force de me battre au quotidien.

A mon professeur encadreur, le **Prof. Dr. Silvia SCHINTKE** qui est pour moi un vrai modèle professionnel de par ses conseils, sa confiance, sa motivation, son attention, sa disponibilité et sa patience. Je n'aurais pas espérer mieux comme encadreur.

Mes remerciements vont également à **Monsieur Christophe GREPPIN** pour sa disponibilité, ses conseils techniques et encouragements. A tous mes professeurs du département TIC qui ont su m'encadrer académiquement pendant ces années.

Je remercie **Madame Jocelyne GUICHARD** pour la relecture de mon rapport.

Ensuite, je tiens à remercier que tous mes collègues, amis et camarades qui de près ou de loin m'ont aidée et soutenue tout au long de ces années. J'ai cité **Thibaut, Kimberley, Nelson, Ursule, Sandrine, Jacques, Créscence, Sylver et Marie-pascale**.

Cahier des charges

Objectifs

Le but de ce projet est de développer une base de données à accès restreint permettant la collecte de données cliniques auprès d'enfants ayant subi des procédures chirurgicales. Elle permettra aussi d'améliorer le traitement de la douleur. Ce projet adresse également la récupération et l'exportation des données vers des outils informatiques spécialisés.

Déroulement

Ce travail comprend :

- Réunions à la HESAV pour identifier des paramètres ou signaux à inclure dans la base de données et des fonctionnalités à implémenter pour leur analyse.
- Recherche sur les formats de données à traiter. Recherche des formats usuels ou standards pour les bases de données médicales/cliniques ainsi que recherche sur des aspects de sécurité pour des bases de données dans le domaine clinique/médical afin de pouvoir en tenir compte.
- Développement de la structure/architecture de la base de données.
- Prioritisation de l'implémentation des données et des fonctionnalités.
- Anticipation de l'automatisation de la lecture de données.
- Programmation avec des logiciels libres de licences.
- Test-runs d'analyse avec la HESAV pour quelques données réelles.
- Proposition d'amélioration et/ou d'autres paramètres à mesurer.
- Documentation technique, rapport et défense du projet.

Livrables

Les éléments livrables seront les suivants :

1. Une documentation contenant :
 - "État de l'art" des DBs¹ et des DBs dans le domaine médical (aspects techniques de développement d'une DB et formats médicaux).

1. DB : Database (Base de données)

- Spécification du cahier des charges techniques de la base de données ("analyse des besoins du client").
 - Choix techniques (pour le stockage et choix de composants et de logiciels ("solutions retenues").
 - Design et développement du prototype de l'application de collecte des données ("la DB réalisée/programmée/implémentée, interface utilisateur").
 - Tests et résultats ("tests de saisie et de lecture").
 - Rédaction du mode d'emploi.
2. Une structure dynamique de la base de données implémentée.
 3. Une application implémentant les améliorations s'il a été possible de les effectuer.

Table des matières

Préambule	v
Authentification	vii
Dédicaces	ix
Remerciements	xi
Cahier des charges	xiii
Table des matières	xvii
1 Introduction	1
2 État de l’art sur les bases de données	3
2.1 Les bases de données	3
2.2 Types de bases de données	3
2.2.1 Avantages des bases de données	4
2.2.2 Inconvénients des bases de données	4
2.3 Bases de données dans le domaine médical	5
2.4 Rôle des bases de données en médecine	5
2.5 Exemples de bases de données médicales de pédiatrie en Suisse et en Europe	5
2.5.1 SwissNeoNet	5
2.5.2 Pedsnet	6
2.6 Formats des données médicales	6
2.6.1 Common Data Model : CDM	6

3	Analyse des besoins du client	8
3.1	Liste des données médicales	10
4	Choix Techniques	12
4.1	Base de données	12
4.1.1	MariaDB	13
4.2	Backend	14
4.2.1	Spring Boot	15
4.3	Frontend	15
4.3.1	Angular	17
4.4	Déploiement : Docker	18
4.5	Hardware	19
5	Architecture	20
6	Conception de la base de données	23
6.1	Modélisation UML	23
6.2	Les tables	24
6.2.1	Relation entre les tables	27
6.3	Importation des données du client vers la base de données	32
6.4	Exportation des données de la base de données sous forme de documents Excel	39
6.5	Transformation des données sous le format CDM	40
6.5.1	Processus	40
6.5.2	Transformation	40
7	Implémentation	43
7.1	Implémentation du backend	43
7.1.1	Entity	44
7.1.2	Controller	45
7.1.3	Services	45
7.1.4	Repository	46
7.1.5	Mapper	46
7.1.6	Sécurité	46

7.1.7	Configuration	50
7.2	Implémentation du frontend	51
8	Tests et résultats	59
8.1	Importation des données depuis le fichier Excel du client (cf ANNEXE video_importation)	59
8.2	Sécurité des données (cf ANNEXE video_sécurité)	59
8.3	Les fonctionnalités	59
8.3.1	Scénario (cf ANNEXE video_fonctionnalites)	60
9	Conclusion	61
	Bibliographie	63
A	Diagramme de Gantt	67
B	Journal de travail	69
C	Concordance des noms	71

Chapitre 1

Introduction

Nous manipulons de nos jours une grande quantité d'informations que l'on appelle communément en informatique des données. Les bases de données permettent de mieux organiser ces données, de les stocker afin qu'elles soient facilement accessibles. Elles sont utilisées dans de nombreux domaines de la vie quotidienne, notamment dans les banques, dans le médical pour la gestion des patients et du personnel de santé, dans la conception des applications de vente en ligne, dans les magasins commerciaux pour suivre les informations au sujet des clients, des stocks, des employés ou dans la comptabilité. Notre travail interviendra dans le domaine médical, plus précisément en chirurgie pédiatrique, où nous serons appelés à développer une structure de base de données pour une institution hospitalière basée en Ukraine.

Les progrès récents dans la correction chirurgicale des malformations et les soins intensifs post-opératoires des bébés nés à terme et prématurés améliorent le taux de survie des enfants. Cependant, les interventions chirurgicales ainsi que de nombreuses procédures pré- et post-chirurgicales restent douloureuses. Cela nécessite une évaluation fiable et une gestion adéquate de la douleur péri- et post-chirurgicale pour minimiser les effets négatifs à court et à long terme sur le développement du nourrisson. Les effets à court terme incluent une diminution de la stabilité physiologique, comme une accélération du rythme cardiaque et une diminution du rythme respiratoire. Les effets à long terme incluent la douleur chronique, des seuils de douleur modifiés et le développement neuro-cognitif, en particulier des altérations de la structure cérébrale, du comportement et des capacités cognitives. Ces effets à distance seraient présents chez les enfants d'âge scolaire et persisteraient jusqu'à l'âge adulte.¹

Pour limiter ces effets, l'évaluation de la douleur est particulièrement problématique chez les nouveau-nés et les très jeunes enfants en raison de leur incapacité à communiquer leur ressenti verbalement. Des changements de développement rapides, en particulier dans les systèmes nerveux central et périphérique à ces âges compliquent encore l'évaluation et la prise en charge de la douleur. Il est généralement admis que pour les raisons ci-dessus, le problème de l'évaluation de la douleur et de la prise en charge adéquate dans ces catégories

1. Tiré du descriptif proposé par la HESAV

de patients reste non résolu. Il existe une forte demande de développement de nouvelles techniques de mesure de la douleur à des fins cliniques chez les enfants, surtout chez les nouveau-nés. Pour établir les meilleures pratiques d'évaluation de la douleur chez les patients ayant une capacité de communication limitée et pour déterminer quelles stratégies de gestion de la douleur améliorent les résultats des patients, nous allons développer une structure de base de données permettant de réaliser la collecte, le stockage et une analyse approfondie des paramètres cliniques caractérisant les conditions des nouveau-nés et des jeunes nourrissons pendant les stades pré, intra et postopératoires de traitement et de réadaptation.

Ce travail comporte ainsi l'état de l'art des bases de données en général et dans le domaine médical en particulier (Chapitre 2), l'analyse de la pratique actuelle de documentation des données et les besoins du mandant (Chapitre 3), le choix des solutions techniques basées sur ces besoins (chapitre 4), la définition de l'architecture adéquate pour la base de données (Chapitre 5), le processus d'implémentation de cette dernière (Chapitre 6), l'implémentation de l'application permettant la collecte (Chapitre 7) et le test de la solution implémentée (Chapitre 8).

Chapitre 2

État de l'art sur les bases de données

Dans ce chapitre nous aborderons les bases de données du point de vue général et dans le domaine médical en particulier. Nous étudierons les bases de données en détail avec leurs avantages et inconvénients, puis nous analyserons l'évolution technique dans le domaine médical

2.1 Les bases de données

Les bases de données jouent un rôle croissant dans les système d'informations, qu'il s'agisse d'applications intranet, e-commerce ou traditionnelles. Elles se définissent comme une collection de données ou d'enregistrements gérés par un système de gestion de base de données (SGBD). Le SGBD est un système logiciel qui permet de stocker et d'organiser efficacement les données spécifiques dans une grande masse d'informations. Les données pourront être supprimées, ajoutées ou modifiées en utilisant un langage de requêtes standards¹.

2.2 Types de bases de données

Il existe plusieurs types de bases de données en informatique. On distingue notamment :

- **Les bases de données hiérarchiques** qui permettent de structurer les informations de façon hiérarchique.
- **Les bases de données en réseaux** qui permettent de créer de multiples liens entres les informations.
- **Les bases de données relationnelles** qui sont constituées de tableaux, dans lesquels les données sont classées par catégorie. Les tableaux permettent d'accéder et de réorganiser les données de façon différente.
- **Les bases de données orientées objet** qui sont identiques aux bases de données relationnelles, sauf qu'elles sont basées autour d'objets et non d'actions.

1. Langage informatique utilisé pour accéder aux données d'une base de données.Exemple : SQL, MDX, SPARSQL, XQuery

- **Les bases de données NoSQL** qui sont utiles pour les larges ensembles de données distribuées et le big data.
- **Les bases de données graphiques** qui sont un type de base de données NoSQL utilisant la théorie des graphes pour manipuler les données.

2.2.1 Avantages des bases de données

Une base de données présente les avantages suivants :

- Elle permet de gérer efficacement et facilement une grande quantité d'informations. Ce qui permettra d'accomplir aisément et beaucoup plus rapidement un nombre de tâches auparavant longues et fastidieuses.
- Elle facilite la tâche la plus laborieuse qui est d'entrer les données, ceci par l'utilisation des formulaires. Ces formulaires augmentent la vitesse de travail et diminuent autant que possible les erreurs.
- Elle facilite les recherches en permettant de faire des tris, par exemple le tri des patients selon un diagnostic précis, le tri des employés dans un service défini, sur demande.
- Elle permet de résoudre des problèmes, de répondre à des questions ou de prendre des décisions. Par exemple elle permet de comprendre les analyses des patients et de faire des prescriptions au vu des résultats.
- Elle fournit une sécurité des données en les protégeant précieusement contre tout accès non autorisé. Les données seront accessibles uniquement par les utilisateurs autorisés, grâce à une authentification appropriée telle qu'un login et un mot de passe.
- Elle permet à plusieurs utilisateurs d'afficher les données en même temps.

2.2.2 Inconvénients des bases de données

Une base de données présente les inconvénients suivants :

- Elle nécessite un matériel coûteux pour une grande quantité de données à gérer. Pour faire tourner le logiciel de gestion des bases de données, l'on a besoin d'un processeur doté d'une grande vitesse et d'une grande taille de mémoire. Ces ressources matérielles sont coûteuses.
- Elle demande beaucoup de temps, et d'expertise pour la conception et la maintenance qui peuvent entraîner des coûts. Pour créer une base de données, l'on a besoin d'un programmeur SQL et d'un administrateur de base de données pour la maintenir une fois qu'elle est construite.
- Les dommages liés à la base de données impliquent la défaillance des programmes l'utilisant.

Besoin	Coût en CHF
Ordinateur avec Ram 8Gb et DDD 500Go	1000.-
Programateur	4500 - 7000.- par mois
administrateur	4500 - 7000.- par mois

TABLE 2.1 – Coût estimatif des besoins pour l’implémentation d’une base de données

2.3 Bases de données dans le domaine médical

La généralisation de l’informatisation des systèmes d’information pour la gestion des données de santé et l’accès à l’information depuis les différents systèmes comportent des enjeux majeurs pour le monde médical. En effet, la multiplication de sources de données médicales a fait naturellement naître de nouveaux espoirs en termes d’utilisation de cette masse d’informations. D’un point de vue médical, la gestion de bases de données facilite non seulement la gestion des données mais aussi leurs possibilités d’exploitation.

2.4 Rôle des bases de données en médecine

En médecine, l’utilisation d’un système de bases de données permet de soutenir la prise de décision concernant la gestion et les soins d’un patient. Elle permet entre autre aussi de stocker une quantité d’informations sous formes de banques de données afin de les réutiliser à des fins de statistiques ou à des fins analytiques. Dans notre travail, elle permettra de stocker les paramètres cliniques collectés afin de les analyser, d’en faire des diagnostics dans le but d’améliorer l’état de santé des nourrissons ayant subi des interventions chirurgicales.

2.5 Exemples de bases de données médicales de pédiatrie en Suisse et en Europe

2.5.1 SwissNeoNet

<https://www.swissneonet.ch>

SwissNeoNet est un groupe suisse de néonatalogie fondé en 1995. Il possède une grande base de données permettant de collecter des données comprenant des informations sur les soins et les résultats des nouveau-nés à hauts risques. La collecte des données est contrôlée en vue de la maîtrise de la couverture de la population, l’exhaustivité de l’ensemble des données, la plausibilité et la fiabilité. Elle recueille aussi dans sa base de données les informations sur l’infrastructure des unités évaluées par le CANU (Comité d’Accréditation des Unités de Néonatalogie).



FIGURE 2.1 – Swissneonet

2.5.2 Pedsnet

<https://pedsnet.org>

Pedsnet est un organisme international comprenant des hôpitaux, des organisations de soins de santé, de chercheurs, de cliniciens, de patients et de familles qui mène des recherches pour améliorer la santé et la vie des enfants. C'est aussi un système de santé pédiatrique dédié à la découverte et la mise en oeuvre de nouvelles façons de fournir les meilleurs soins et d'améliorer les résultats des soins. Il possède une très grande base de données contenant des informations sur une grande population d'enfants, collectées sur de nombreux sites et systèmes sources.



FIGURE 2.2 – Pedsnet

2.6 Formats des données médicales

Introduit en 1992, le format de données EDF est devenu la norme pour les enregistrements de signaux. Ce format européen de données (EDF) est un format simple et flexible pour l'échange et le stockage de signaux biologiques et physiques multicanaux [1]. Les fichiers EDF permettent de stocker des enregistrements biomédicaux tels que l'électroencéphalogramme (EEG), l'électromyogramme (EMG), l'électrocardiogramme (ECG) ou la polysomnographie (PSG).

2.6.1 Common Data Model : CDM

L'organisme PEDSnet a établi un modèle commun de données pédiatriques, que l'on appelle PCDM (PEDSnet Common Data Format), pour le stockage des données PEDSnet. L'utilisation d'un CDM interne permet à PEDSnet d'ajouter rapidement des domaines de données ou des éléments de données nécessaires aux enquêteurs pédiatriques. Le PEDSnet CDM est basé sur OMOP (Observational Medical Outcomes Partnership), un modèle de données communes issu de l'OHDSI (Observational Health Data Sciences and Informatics) et se concentre sur la normalisation terminologique, résultant en l'utilisation de terminologies standard communes [5]. Pour exporter les données, nous utiliserons les terminologies de ce modèle afin que la base de données développée puisse être compatible au PCDM. Par exemple selon le PCDM, un patient est représenté par la table **Person**, qui comprend des attributs comme décrit dans le tableau suivant :

Champs	Contrainte Not Null	Type des données
<i>person_id</i>	Oui	Integer
<i>person_source_value</i>	Non	Varchar
<i>pn_gestation_age</i>	Non	VarChar
<i>race_concept_id</i>	Oui	Integer

TABLE 2.2 – Convention PCDM

En résumé, il existe plusieurs types de bases de données permettant de stocker les informations. Elles interviennent dans plusieurs domaines de la vie courante telles que la médecine, la biométrie, les banques, la vente, l'agriculture ou l'astronomie. Nous avons dans cette première partie abordé les différents types de bases de données existantes, à savoir les bases de données hiérarchiques, en réseaux, relationnelles et orientées objet. Par la suite, nous avons évoqué les bases de données dans le domaine médical, en citant quelques bases de données existantes en Suisse (Swissneonet) et en Europe (Pedsnet). Puis, nous avons présenté le format des fichiers médicaux EDF qui est le format des fichiers de données européen permettant de lire les enregistrements issus des moniteurs médicaux². Pour terminer nous avons présenté le Common Data Model qui est le modèle de format de base de données requis pour une base de données clinique.

2. Moniteur médical : Typiquement un appareil qui enregistre des fonctions vitales dans un contexte médical.

Chapitre 3

Analyse des besoins du client

Résumé du problème

L'institution hospitalière pour laquelle nous réalisons ce projet, utilise des fichiers Excel comme moyen de stockage d'informations relatives à un patient. Les informations sont collectées sur des formulaires en papier puis stockées dans des feuilles de calcul Excel. La recherche de l'information dans ces documents n'est pas aisée et demande beaucoup de temps. Chaque fois qu'un patient est admis dans l'institution une grande quantité d'informations personnelles et confidentielles est collectée et enregistrée.

Problématique

L'évaluation de la douleur est un réel problème chez les nourrissons et les très jeunes enfants en raison de leur incapacité à communiquer leur ressenti verbalement. Par exemple lorsque ces derniers subissent une intervention chirurgicale, il n'est pas aisé d'évaluer si les pleurs sont des cris de douleur ou de faim. Le problème de l'évaluation de la douleur et de la prise en charge adéquate dans ces catégories de patients reste non résolu et il existe une forte demande de développement de nouvelles techniques de mesure de la douleur à des fins cliniques chez ces derniers. La solution existante employée par l'institut hospitalier est la collecte manuscrite des données avec des formulaires et l'utilisation de feuilles de calcul Excel pour stocker les informations relatives à chaque patient. Chaque colonne du fichier Excel représente ainsi le type de données à enregistrer et chaque ligne, les données relatives à un nouveau-né précis.

Solutions possibles

La réalisation de la collecte, du stockage et de l'analyse approfondie des paramètres cliniques qui caractérisent les conditions des nouveau-nés pendant les stades de l'opération, du traitement et de réadaptation, sera possible avec le développement d'une base de données. La base de données sera dédiée avec un accès restreint et pourra permettre la collecte

de données cliniques telles que le nom du nourrisson, son poids, les informations sur sa mère, sa taille, son âge gestationnel, son taux de glucose, son pouls, sa fréquence respiratoire ou sa pression artérielle. Cette solution permettra aux personnels soignants de définir un traitement, de récupérer des informations et d'exporter des données vers des outils informatiques spécialisés. Nous allons réaliser un cahier des charges afin de répondre aux besoins de l'institution hospitalière.

Cahier des charges

Pour développer la base de données afin d'appliquer la solution citée plus haut, nous allons procéder selon le cahier des charges suivant :

1. Visites à la Haute Ecole de Santé de Vaud. Elément aborder :
 - Discussion sur la problématique que soulève le projet.
 - Présentation des solutions existantes.
 - Définition des attentes du clients.
 - Présentation des solutions possibles.
 - Fourniture des données existantes à titre de test d'insertion.
2. Compréhension approfondie des besoins et de la problématique du client. Ceci par une présentation de la solution existante, des attentes finales et une série de questions et réponses.
3. Analyse de la solution existante et des besoins
4. Elaboration d'un diagramme de Gantt pour la gestion visuelle du projet et la structuration des tâches.
5. Recherche sur les différents types de bases de données et choix de la base de données adéquate pour la résolution de la problématique.
6. Recherche et étude des bases de données médicales existantes, ainsi que du format des fichiers et des données cliniques.
7. Etude, comparaison et choix des différentes solutions techniques permettant d'implémenter la base de données.
8. Définition de la structure de la base de données pour la collecte et le stockage des données ci-après :
 - Données sur le nouveau-né : nom (ID), date de naissance, âge gestationnel, croissance, poids, score d'Apgar, diagnostic, maladies concomitantes et caractéristiques de la période périnatale, présence de convulsions, arythmies cardiaques et état neurologique.
 - Données sur les antécédents familiaux : nom de la mère, âge, maladies maternelles de grossesse, antécédents obstétricaux, accouchement, type d'accouchement, complications de la naissance et période post-partum.
 - Données en période pré-opératoire, péri-opératoire et post-opératoire.

- Données du moniteur : enregistrement de l'activité électrique du cerveau (surveillance EEG), enregistrement quotidien de la fréquence cardiaque et des paramètres de l'état fonctionnel (chemin vers le fichier d'enregistrement).
9. Définition des tables, des attributs, des types, des relations à inclure dans le modèle conceptuel.
 10. Création du modèle conceptuel permettant de représenter le problème par la modélisation du schéma UML de la base de données.
 11. Traduction du modèle conceptuel en modèle logique.
 12. Implémentation dans un SGBD (Système de Gestion de base de Données), à partir du modèle logique.
 13. Implémentation suivant le format du Pedsnet Common Data Model.
 14. Importation des données existantes dans la base de données.
 15. Implémentation du backend permettant de créer des services qui interagiront avec la base de données.
 16. Définition des accès par la spécification des utilisateurs autorisés à interagir avec la base de données, à savoir le médecin-chef, les médecins, et les infirmières en charge du patient.
 17. Récupération des informations sur la modification des données, à savoir l'utilisateur qui effectuera l'action et la valeur modifiée.
 18. Implémentation du front-end représentant l'interface utilisateur.
 19. Déploiement de la solution avec docker sur un serveur ayant une adresse IP privée.

3.1 Liste des données médicales

Nous avons reçu du client un document Excel (nous ne pouvons pas le fournir en ANNEXE car il est confidentiel) contenant des données collectées sur différents enfants. Les données du tableau Excel étant brutes nous les avons réorganisées afin de réaliser l'importation. Ci-dessous un tableau représentatif de ces données, leurs significations ainsi que leurs unités de grandeur.

Données	Unités
ID Neo (Number of case history)	No unit
Surname	No unit
Gender	No unit
Age	mois
Weight	g
Hight	inches
Diagnosis	No unit
Starting date of observation	yy.mm.dd
Chirurgical intevention date	yy.mm.dd
Anesthesia type	No unit
HR (heart rate)	beats per minute
BR (breasing rate)	cycles per minute
Systolic blood pressure	mmHg
Diastolic blood pressure	mmHg
FiO2 (O2 respiratory mixture)	pourcentage
SpO2 (Saturation O2)	pourcentage
Hb (hemoglobin)	g/L
Ht (hematocrit)units*100	pourcentage
Er (red blood cells)	(units* 10^{12}) per L
Leik (white blood cells)	(units * 10^9) per L
Glucose	mmol per L
K (potassium)	mmol per L
Na (sodium)	mmol per L
Ca (calcium)	mmol per L
Pv O2 (partial oxygen)	mmHg
PvCO2 (partial carbon dioxide)	mmHg
BE (acid-base balance)	mmol per L
pH	No unit

TABLE 3.1 – Liste des données reçues

Chapitre 4

Choix Techniques

Dans ce chapitre, nous allons présenter les différentes solutions techniques permettant de développer notre base de données et notre application de collecte des données, ceci en comparant des solutions possibles puis en justifiant nos choix.

4.1 Base de données

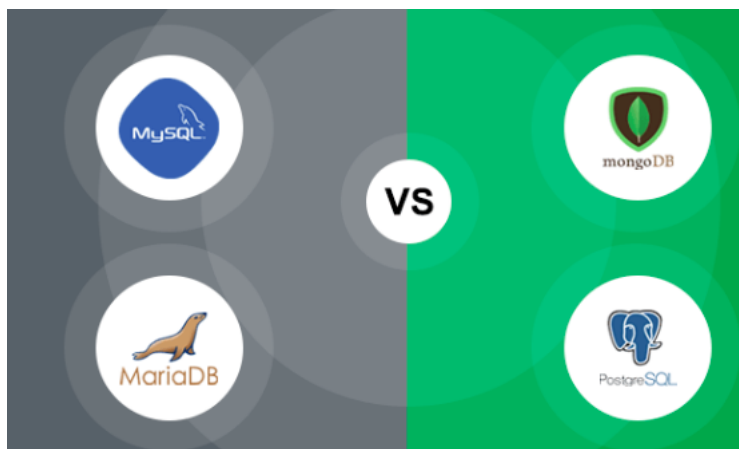


FIGURE 4.1 – MySQL vs MariaDB vs PostgreSQL vs MongoDB

La gestion d'une base de données se fait par un SGBD (Système de Gestion des Bases de données). Le SGBD est un logiciel qui permet le stockage de grandes quantités de données persistantes et qui facilite l'accès multi-utilisateurs aux données de façon efficace, fiable et sécurisée. Afin de déterminer quel SGBD serait le plus intéressant à utiliser pour créer la base de données, nous avons fait une étude comparative de quelques SGBD à savoir MySQL, MariaDB, PostgreSQL et MongoDB que nous avons classé dans le tableau ci-dessous en fonction de cinq critères : la fonction principale, le type de base de données, la structure des données, la documentation et le type de licence.

SGBD	MySQL	MariaDB	PostgreSQL	MongoDB
Fonction principale	Permet de spécifier et de gérer des données structurées suivant les principes de l'algèbre relationnelle	Idem que MySQL car c'est le fork de ce dernier	Utilisation des objets définis par l'utilisateur et d'approche pour créer des structures de données plus complexes	Base de données spécialisée, basée sur une architecture non relationnelle de stockage de documents
Type de base de données	Relationnelle	Relationnelle	Relationnelle	Non relationnelle
Structure des données	SQL	SQL	SQL, Objet-relationnel	NoSQL, document orienté
Documentation	Moins documenté	Plus documenté	Moins documenté	Plus documenté
Licence	Propriétaire	GNU Generally Public Licence	Open-source	SSPL

TABLE 4.1 – Tableau comparatif de quelques SGBD

4.1.1 MariaDB

Après avoir fait l'analyse comparative, nous avons choisi d'utiliser MariaDB comme gestionnaire de la base de données. MariaDB est un fork¹ de MySQL. Il possède une licence gratuite, comparé à MySQL qui a actuellement plusieurs fonctionnalités payantes. MariaDB utilise le langage SQL que nous avons étudié. Il supporte les langages de programmation tels que le C, le C++ et le Java que nous avons appris à maîtriser pendant nos cours de programmation informatique.



FIGURE 4.2 – MariaDB

MariaDB utilise le type de base de données relationnel qui est un type facile d'utilisation et intuitif. De plus sa structure peut être modifiable sans avoir à changer l'application en elle-même. Il garantit que les attributs ne soient pas dupliqués. Dans ce type, les données sont rangées en table, dans des lignes et des colonnes, ce qui facilite leur accessibilité. Les tables contiennent donc toutes les informations sur les relations entre les données. Une ligne peut représenter un patient et une colonne des attributs qui se rapportent à ce dernier tels que le nom, prénom, la taille, l'âge ou la pression artérielle.

Nous utilisons cependant la version de MariaDB 10.4.10 compatible avec MacOS Catalina version 10.15.4.

1. fork : Logiciel créé à partir du code source d'un autre logiciel.

4.2 Backend

Le **backend** est la partie immergée de notre application de collecte des données. Il communiquera avec la base de données afin de faire toutes les requêtes nécessaires et envoyer les réponses à l'interface utilisateur.

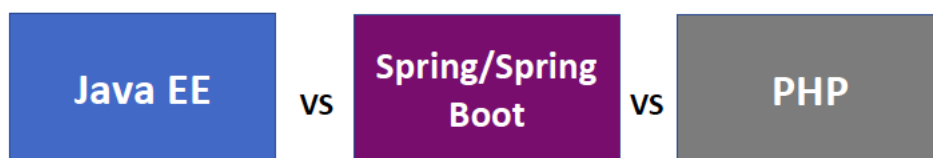


FIGURE 4.3 – Java EE vs Spring Boot vs PHP

Pour mettre en place notre backend, nous avons eu le choix entre JavaEE, Spring et PHP. L'alternative PHP a été écartée afin de préférer une technologie basée sur le langage Java, langage que nous avons étudié pendant notre formation. Le framework² Spring fournit un support d'infrastructure complet pour le développement d'applications et le framework Java EE est un outil puissant pour créer les applications complexes et larges. Le tableau comparatif ci-dessous des frameworks JavaEE et Spring pour l'implémentation du backend nous aidera dans notre choix.

Framework	JavaEE	Spring/Spring Boot
Architecture	Basé sur un cadre architectural tridimensionnel, à savoir les niveaux logiques, les niveaux client et les niveaux de présentation	Il est basé sur une architecture en couches qui comprend de nombreux modules. Ces modules sont fabriqués au-dessus de son conteneur principal
Langage	Utilise un langage orienté objet de haut niveau qui a un certain style et une syntaxe	Pas de modèle précis de programmation
Interface	Possède généralement une interface utilisateur graphique créée à partir des API Project Swing ou Abstract Window Toolkit	Syntaxe identique partout - indépendamment d'un IDE ou d'un compilateur
Injection de dépendance³	Utilise l'injection de dépendance	Utilise l'injection de dépendance
Structure	Peut être basé sur le Web ou non	Basé sur au moins 20 modules
Vitesse	Assez rapide	Moins rapide
Documentation	Bien documenté	Bien documenté
Licence	Oracle	Apache

TABLE 4.2 – Tableau comparatif de Java EE vs Spring/Spring Boot [3]

2. **Framework** : Ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel [6]

4.2.1 Spring Boot

Après analyse du tableau comparatif ci-dessus, nous avons choisi d'utiliser l'extension du framework Spring qui est **Spring boot** et qui est basé sur toutes les fonctionnalités de ce dernier. Spring boot permet d'écrire beaucoup moins de code standard en utilisant des annotations et de la configuration XML (Extensible Markup Language), ce qui augmente la productivité lors de l'implémentation. On utilise dans le framework Spring Boot, les applications de Spring suivantes :



FIGURE 4.4 – Springboot

- **Spring JDBC (Java Database Connectivity)** pour l'accès à la base de données.
- **Spring ORM (object-relational mapping)** pour le mapping d'objets relationnels.
- **Spring Data** pour la communication avec la base de données.
- **Spring Security** pour la sécurité.

Ce qui motive davantage notre choix est la connaissance que nous avons de ce framework puisqu'il fut l'objet d'un cours d'apprentissage pendant notre formation et nous l'avons utilisé pour réaliser plusieurs projets de programmation.

4.3 Frontend

Le frontend est la partie permettant la conception de l'interface utilisateur. C'est la partie visible de l'application. Pour réaliser le frontend d'une application, l'on utilise les langages suivants :

- **CSS (Cascading Style Sheets)** qui est un langage de feuilles de style permettant d'appliquer du style tel que la couleur afin de rendre les pages attrayantes.
- **l'HTML (HyperText Markup Language)** qui est un langage descriptif permettant de construire et d'afficher les pages.
- **JavaScript** qui est un langage de programmation de scripts permettant de créer les pages web interactives. Il intègre facilement le langage HTML.

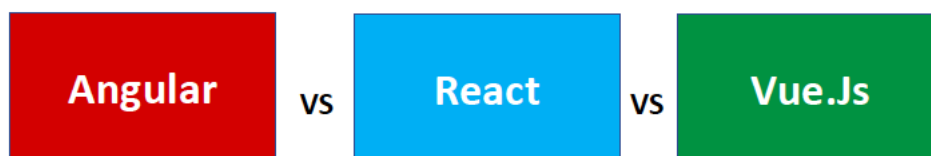


FIGURE 4.5 – Angular vs React vs Vue.js

Pour la partie frontend de notre application de collecte de données, nous avons le choix entre Thymeleaf⁴ ou l'un des fameux framework frontend que sont Angular, React et Vue.js. La deuxième approche nous permettait de davantage découpler le backend du frontend. Ci-dessous un tableau comparatif des frameworks Angular, React et Vue.js.

Framework	Angular	React	Vue.js
Fonctionnalité	Permet de construire des interfaces utilisateurs et des applications web SPA (Single-Page Application) ⁵	Permet de faciliter la création d'application web SPA, via la création de composants dépendant d'un état et générant une page HTML à chaque changement d'état	Permet de développer des pages web même les plus complexes
Langage	TypeScript	JavaScript	JavaScript
Communauté	Une grande communauté de développeurs	Communauté de développeurs Facebook	open-source sponsorisé par le crowdsourcing ⁶
Développeur	Google	Facebook et Instagram	Evan You (projet Github)
Facilité d'apprentissage	Difficile	Difficile	Facile
Documentation	assez bien documenté	assez bien documenté	très bien documenté
Licence	Open source MIT	Open source MIT	Open source MIT

TABLE 4.3 – Tableau comparatif de Angular vs react vs Vue.js

4. Thymeleaf : moteur de template Java pour HTML

4.3.1 Angular

Après analyse du tableau comparatif ci-après, les frameworks Angular et React paraissent trop difficiles à appréhender au vu du peu de documentation à disposition et de la difficulté d'apprentissage. Cependant la connaissance du framework Angular est de plus en plus demandé sur le marché de l'emploi mondial et Suisse. Selon Google Trends, l'on remarque que Angular est le framework frontend le plus en utilisé dans le monde et aussi en Suisse. Ci-dessous des graphes obtenus par Google Trends qui est un site d'analyses et de statistiques.

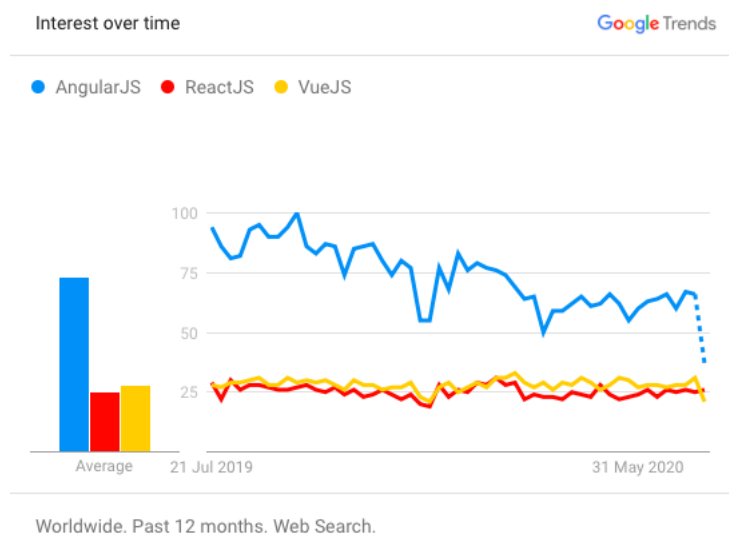


FIGURE 4.6 – Google Trends : Angular vs React vs Vue.js dans le monde [4]



FIGURE 4.7 – Google Trends : Angular vs React vs Vue.js dans le monde par pays [4]

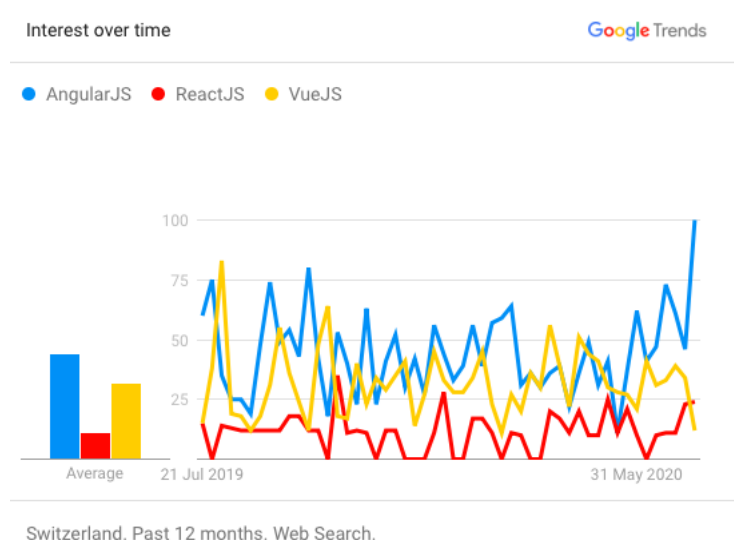


FIGURE 4.8 – Google Trends : Angular vs React vs Vue.js en Suisse [4]

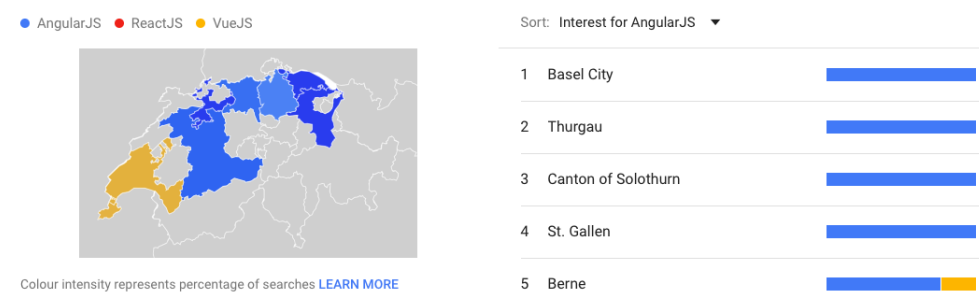


FIGURE 4.9 – Google Trends : Angular vs React vs Vue.js en Suisse par canton [4]

Au vu de cette tendance nous avons décidé d'utiliser Angular. Ce fut un challenge de se pencher sur ce framework étant donné que le typescript est un langage de script que nous n'avons pas appris pendant notre formation. Nous l'avons donc appris pendant les mois à disposition pour la réalisation de ce travail afin d'être compétitif sur le marché de l'emploi. Version d'Angular utilisé : **9.1.9**

4.4 Déploiement : Docker

Nous avons choisi de déployer les services de notre base de données dans un conteneur Docker. **Docker** est un outil conçu pour faciliter la création, le déploiement et l'exécution d'applications à l'aide de conteneurs. Les conteneurs nous permettent de mettre en paquet la base de données avec toutes les parties dont elle a besoin, telles que les ports d'écoutes, les dépendances, et de la déployer en un seul paquet. Les services que nous avons choisi de déployer avec Docker sont le SGBD MariaDB et l'administrateur de base de données

PhpMyAdmin, ce qui permettra de les lancer dans un terminal avec l'unique commande *docker-compose up*

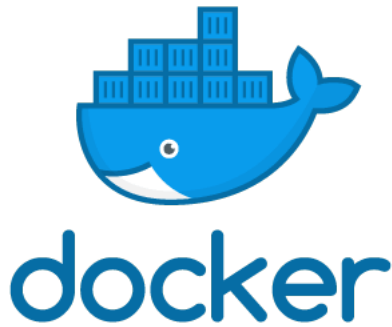


FIGURE 4.10 – Docker

4.5 Hardware

Pour Stocker notre base de données nous avons choisi d'utiliser un serveur hébergé à la HEIG-VD avec une adresse IP public. Les caractéristiques de notre serveur sont les suivantes :

- **Adresse IP** : 10.192.75.97
- **User** : heiguser
- **Port d'entrée et sortie** : 80 et 443
- **Ram** : 4Go
- **Disque dur** : 100Go

Chapitre 5

Architecture

Ce chapitre a pour but de présenter l'architecture globale de notre projet. Cette architecture permet ainsi d'organiser des différents éléments du projet (logiciels et/ou matériels et/ou humains) et les relations entre ces éléments. Elle fait suite à un ensemble de décisions stratégiques prises durant la conception de ce dernier.

La conception de notre architecture suit le principe de l'architecture à trois tranches (**3-tier architecture** en langage informatique) qui est une architecture client-serveur dans laquelle la logique de processus fonctionnelle, l'accès aux données, le stockage des données informatiques et l'interface utilisateur sont développés et maintenus en tant que modules indépendants sur des plates-formes distinctes.[9] Cette dernière permet cependant d'avoir une grande flexibilité au niveau du développement de l'application, du fait que les niveaux de conception sont indépendants les uns des autres.

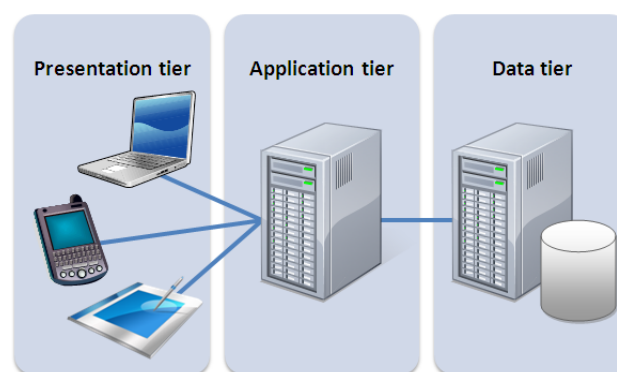


FIGURE 5.1 – Architecture **3-tier**.

La Figure 5.1 est structurée en trois niveaux :

- Le niveau **présentation** est la partie visible par l'utilisateur. Il représente le frontend de l'application.
- Le niveau **application** est la partie intermédiaire. Il est le serveur de l'application

et permet d'effectuer des opérations sur les données.

- Le niveau **data** est la partie de l'architecture permettant d'effectuer les opérations de stockage des données.

L'architecture que nous avons conçue pour élaborer notre projet, calquée sur l'architecture tree-tier est la suivante :

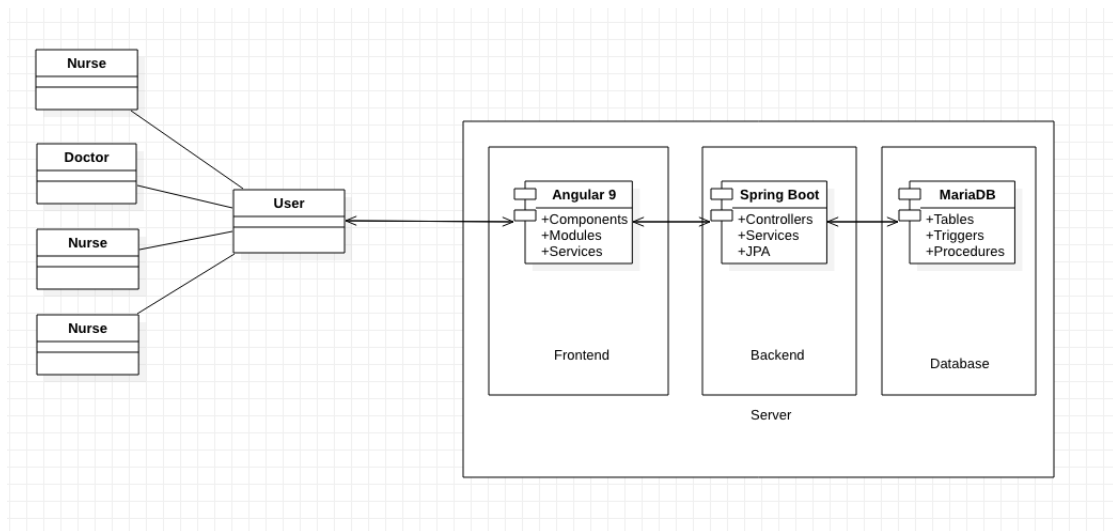


FIGURE 5.2 – Architecture du projet.

Scénario :

- L'utilisateur interagit avec l'application par le navigateur Web qui envoie des requêtes HTTP au serveur.
- L'application récupère les informations de l'API.
- L'API va chercher l'information dans la base de données.

L'architecture illustrée à la **Figure 5.2** est découpée en trois parties :

1. Le backend qui utilise le framework **Spring boot**. Cette partie est liée à la base de données et communique directement avec celle-ci.
 - Elle est chargée de faire des requêtes à la base de données. Faire des requêtes consiste à faire une demande de récupération des données. Cependant l'on peut sans autre manipuler les données avec des requêtes ; par exemple effectuer des insertions, des modifications ou des suppressions de données. Les données proviennent donc d'une ou plusieurs tables.
 - Elle permet également de servir des routes HTTP pour exposer des ressources. Les ressources représentent des collections appartenant à un domaine. Si l'on souhaite par exemple exposer un ensemble de données médicales, afin que le médecin puisse rechercher ou modifier une valeur clinique, la valeur clinique sera considérée comme ressource.

2. Le frontend qui utilise le framework **Angular**. Angular un framework de création d'applications clientes mono-pages (SPA : Single Page Application) à l'aide des langages typescript et HTML. Ce framework utilise la notion de modules et de composants pour fournir une interface réactive et dynamique à un utilisateur. Le lien entre l'utilisateur final et l'application frontend est le navigateur Web.
3. La base de données dont les données se trouve dans le SGBD **MariaDB**. La base de données assure la persistance des données ; c'est-à-dire qu'elle permet de les stocker. Les données sont donc disponibles à tout moment même à l'arrêt de l'application. Etant donné la sensibilité des informations à stocker, la base de données aura un accès restreint.

Chapitre 6

Conception de la base de données

Ce chapitre comporte les étapes nécessaires à la mise en place de notre base de données (database). Après avoir identifié la problématique et les besoins du client, déterminé les choix techniques et défini l'architecture, nous allons modéliser le problème sous forme d'UML, rédiger le script de création de la base de données ainsi que celui de la création des tables, exporter les données reçues du client pour les stocker et enfin exporter les données de la base de données sous forme de fichier SQL ou CSV.



FIGURE 6.1 – Database

6.1 Modélisation UML

La conception d'une base de données passe par la modélisation du problème. Pour cela nous allons utiliser le schéma UML (Unified Modeling Language) qui est utilisé pour spécifier, visualiser, modifier et construire l'architecture nécessaire au bon développement de la base de données. Le schéma UML offre un standard de modélisation pour représenter l'architecture logicielle. Le schéma UML représentatif de notre base est le suivant :

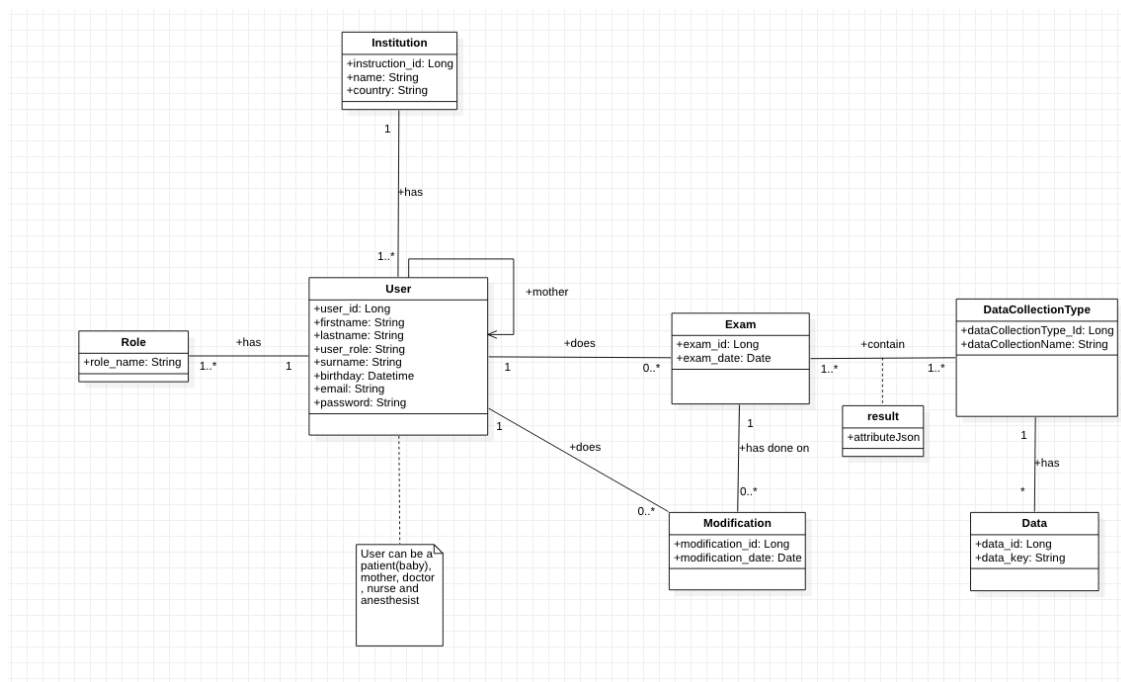


FIGURE 6.2 – Schéma UML

6.2 Les tables

Nous avons utilisé huit tables pour modéliser notre schéma UML. Une table est un ensemble de données organisées sous forme d'un tableau où les colonnes correspondent à des catégories d'informations et les lignes à des enregistrements, également appelés entrées.[11]. Les différentes tables sont les suivantes :

1. La table **Institution** : Cette table représente une institution hospitalière. Elle possède l'attribut *institution_id* qui est l'identification unique de l'institution. L'attribut *name* représente le nom de l'institution et l'attribut *country* représente le pays où se situe l'institution.

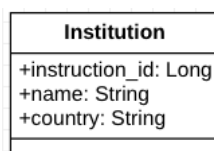


FIGURE 6.3 – Table Institution

2. La table **Role** : Cette table permet de stocker les différents rôles des utilisateurs. Le rôle permet ainsi d'attribuer des droits d'accès à la base de données ainsi qu'autoriser ou restreindre certaines actions dans l'application. Cette table possède l'attribut

role_name qui représente le nom du rôle. Les différents rôle de notre base de données sont : **admin**, **doctor**, **nurse** et **patient**.

Role
+role_id: Long
+role_name: String

FIGURE 6.4 – Table Role

3. La table **User** : Cette table permet de stocker différents utilisateurs tels que l'administrateur, les médecins, les infirmières, les anesthésistes, les patients nourrissons ainsi que leurs mères. L'utilisateur accèdera à l'application par un e-mail et un mot de passe. Seuls les utilisateurs du corps médical ayant les droits d'accès peuvent collecter, modifier, afficher et enregistrer les informations dans la base de données. Cette table possède l'attribut *user_id* qui représente l'identifiant unique de l'utilisateur. Elle contient aussi les attributs *firstname*, *lastname*, *surname*, *birthday* qui représentent les informations personnelles de l'utilisateur. L'attribut *user_role* représente la fonction de l'utilisateur au sein de l'institution et l'attribut *password* représente son mot de passe permettant de se connecter à l'application.

User
+user_id: Long
+firstname: String
+lastname: String
+surname: String
+birthday: Date
+email: String
+password: String
+created_date

FIGURE 6.5 – Table User

4. La table **Exam** : Cette table permet de stocker les différents examens effectués sur un patient. Faire un examen revient à se préparer à faire une collecte ou une modification des données médicales. Cette table possède l'attribut *exam_id* qui représente l'identifiant unique de l'examen, l'attribut *exam_date* qui représente la date à laquelle l'examen a été réalisé, l'attribut *doctor_name* qui représente le nom du médecin ou de l'infirmière qui a réalisé l'examen.

Exam
+exam_id: Long
+exam_date: Date
+doctor_name: String

FIGURE 6.6 – Table Exam

5. La table **Result** : Cette table permet de stocker les résultats d'un examen. Elle possède l'attribut *result_id* qui représente l'identification unique d'un résultat. L'attribut *step* représente les moments de la collecte. Lorsqu'une opération est effectuée sur un patient, les données vitales sont collectées au début de l'opération, pendant

l'opération, pendant le moment traumatique, pendant le moment d'induction et 24 heures après l'opération. Elle possède l'attribut *attribut_json* qui représente l'objet JSON et qui permet de stocker et de transmettre des objets de données constitués de paires clé-valeur. Cette table rend la base de données évolutive car elle permet d'ajouter autant de paires clé-valeur possibles dans l'objet JSON.

result
+result_id_Long
+attribute_json: JSON
+step: String

FIGURE 6.7 – Table Exam

6. La table **DataCollectionType** : Cette table permet de stocker les différentes collections de données. Elle possède un attribut *data_collection_id* qui représente l'identifiant unique de la collection de données et un attribut *data_collection_name* qui représente le nom de la collection. Comme collections de données nous avons :

- Une collection de données liées à l'enfant (childData).
- Une collection de données liées à la mère (motherData).
- Une collection de données liées à une opération chirurgicale (operationData).
- Une collection de données liées aux différents moniteurs (monitorData).

DataCollectionType
+dataCollectionType_Id: Long
+dataCollectionName: String

FIGURE 6.8 – Table DataCollectionType

7. La table **Data** : Cette table permet de stocker toutes les données relatives aux collections de données. Par exemple pour une collection de données **childData**, l'on stockera les données telles que : l'âge, le poids, le genre, les maladies concomitantes, les caractéristiques de la période périnatale, le type de la nutrition, la présence de convulsions, l'arythmies cardiaques ou l'état neurologique. Les données représenteront les différentes clés de notre table basée sur la relation clé-valeur. Cette table possède un attribut *data_id* qui représente l'identifiant unique de la donnée, un attribut *data_key* qui représente la clé et un attribut *data_value* qui représente la valeur associée à la clé. Exemple de relation clé-valeur :

key: Weight
value: 2896 g

Data
+data_id: Long
+data_key: String
+data_value: String

FIGURE 6.9 – Table Data

8. La table **Modification** : Cette table permet d’avoir une trace sur les valeurs modifiées afin de renseigner l’auteur de la modification et l’heure à laquelle cette dernière a été effectuée. Elle possède un attribut *modification_date* qui représente la date de modification des données existantes.

Modification
+modification_date: Date

FIGURE 6.10 – Table Modification

6.2.1 Relation entre les tables

Après avoir réaliser le modèle conceptuel de la base de données par l’UML, nous allons transformer ce dernier en modèle relationnel. Pour mettre en relation les différentes tables nous avons eu à définir les cardinalités¹ puis les clés. Le modèle relationnel nous a permis d’écrire le script de création des tables en langage SQL ci-dessous. La relation entre les tables se fait par une clé étrangère qui est une clé permettant de gérer les relations entre plusieurs tables et garantir la cohérence des données.

1. Description des relations

1. La cardinalité est le nombre de participations d’une entité à une relation

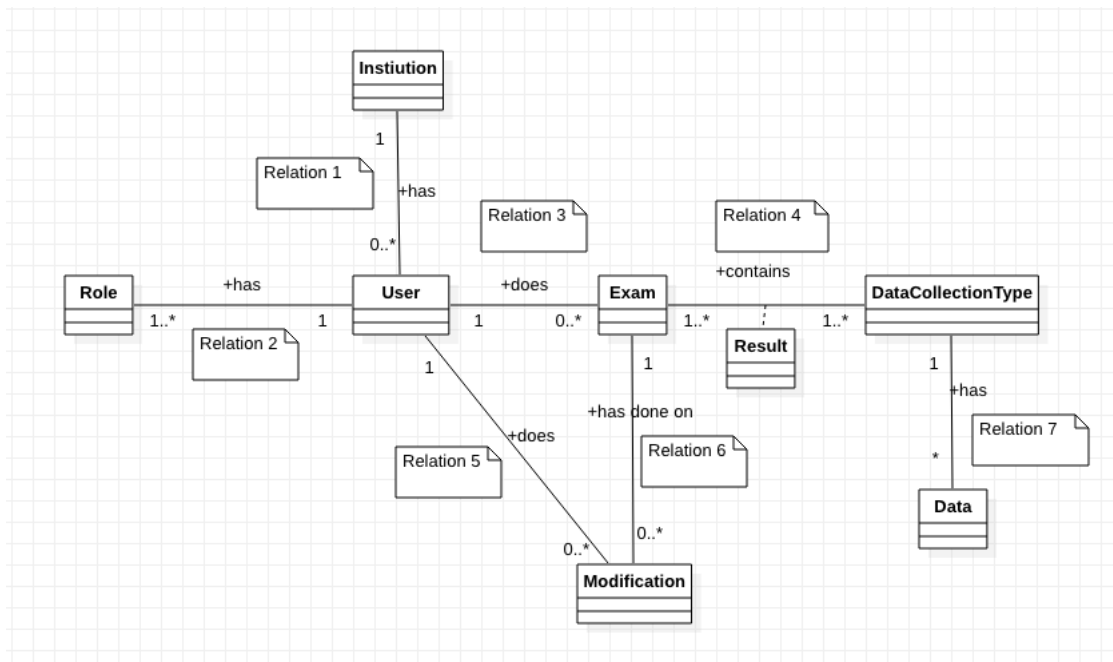


FIGURE 6.11 – Relation entre les tables

La Figure 6.10 est décrite ci-dessus :

- **Relation 1** : une(1) institution peut avoir zéro ou plusieurs(0..*) occupants qui sont les utilisateurs.
- **Relation 2** : un(1) utilisateur peut avoir un ou plusieurs(0..*) rôles.
- **Relation 3** : un(1) utilisateur peut effectuer zéro ou plusieurs(0..*) examens.
- **Relation 4** : un(1) ou plusieurs examens peuvent contenir un ou plusieurs(1..*) types de données.
- **Relation 5** : un(1) utilisateur peut effectuer zéro ou plusieurs(0..*) modifications.
- **Relation 6** : zéro ou plusieurs(0..*) modifications peuvent être effectuées sur un examen(1).
- **Relation 7** : une(1) collection de données contient plusieurs données(*).

2. Script de création de la base de données, des tables et des relations

Code Source : Création de la base de données

```

1 DROP DATABASE IF EXISTS ClinicalChildData; /*supprime la base de données si
   ↳ elle existe*/
2 CREATE DATABASE ClinicalChildData; /*création de notre base de données*/
3 USE ClinicalChildData; /*indique que nous utilisons cette DB*/
4
5 DROP TABLE IF EXISTS user;

```

```

6 DROP TABLE IF EXISTS institution;
7 DROP TABLE IF EXISTS exam;
8 DROP TABLE IF EXISTS dataCollectionType;
9 DROP TABLE IF EXISTS data;
10 DROP TABLE IF EXISTS modification;
11 DROP TABLE IF EXISTS role;
12
13
14 CREATE TABLE IF NOT EXISTS institution ( /*création de la table institution
    ↪ dans la DB*/
15     institution_id BIGINT(20) NOT NULL AUTO_INCREMENT PRIMARY KEY,
16     institution_name VARCHAR(90),
17     country VARCHAR(90)
18 ) ENGINE=InnoDB
19     CHARSET = utf8mb4
20     COLLATE = utf8mb4_unicode_ci;
21
22 CREATE TABLE IF NOT EXISTS role ( /*création de la table role dans la DB*/
23     role_id BIGINT(20) NOT NULL AUTO_INCREMENT PRIMARY KEY,
24     role_name VARCHAR(90) NOT NULL,
25     CONSTRAINT uk_module_role_name UNIQUE (role_name)
26 ) ENGINE=InnoDB
27     CHARSET = utf8mb4
28     COLLATE = utf8mb4_unicode_ci;
29
30 CREATE TABLE IF NOT EXISTS user ( /*création de la table user dans la DB*/
31     user_id BIGINT(20) NOT NULL AUTO_INCREMENT PRIMARY KEY,
32     role_id BIGINT(20),
33     firstname VARCHAR(50),
34     lastname VARCHAR(50),
35     surname varchar(90),
36     email varchar(90) NULL,
37     birthday DATE,
38     created_date DATE,
39     password LONGBLOB NULL,
40     Salt LONGBLOB NULL,
41     mother_id BIGINT(20),
42     institution_id BIGINT(20),
43
44     /*Etablissement des relations à partir des clés étrangères*/
45     CONSTRAINT fk_institution_user
46     UNIQUE KEY uk_module_email (email),
47     FOREIGN KEY (institution_id) REFERENCES institution(institution_id),
48     CONSTRAINT fk_is_mother
49     FOREIGN KEY (mother_id) REFERENCES user(user_id),
50     CONSTRAINT fk_user_role
51     FOREIGN KEY (role_id) REFERENCES role(role_id)
52     ON DELETE CASCADE /*quand on supprime un utilisateur on supprime ses
    ↪ données*/
53
54 ) ENGINE=InnoDB
55     CHARSET = utf8mb4
56     COLLATE = utf8mb4_unicode_ci;
57

```

```

58 CREATE TABLE IF NOT EXISTS exam( /*création de la table exam dans la DB*/
59     exam_id BIGINT(20) NOT NULL AUTO_INCREMENT PRIMARY KEY,
60     user_id BIGINT(20) UNIQUE,
61     exam_date DATE,
62     docter_name varchar(90) DEFAULT NULL,
63     diagnosis varchar(255) DEFAULT NULL,
64     CONSTRAINT `fk_exam_user`
65     FOREIGN KEY(user_id) REFERENCES user(user_id)
66
67 ) ENGINE=InnoDB
68     CHARSET = utf8mb4
69     COLLATE = utf8mb4_unicode_ci;
70
71 CREATE TABLE IF NOT EXISTS data_collection_type(
72     data_collection_id BIGINT(20) NOT NULL AUTO_INCREMENT PRIMARY KEY,
73     data_collection_name VARCHAR(90) NOT NULL UNIQUE
74
75 ) ENGINE=InnoDB
76     CHARSET = utf8mb4
77     COLLATE = utf8mb4_unicode_ci;
78
79 CREATE TABLE IF NOT EXISTS result( /*création de la table result dans la
    ↪ DB*/
80     result_id BIGINT(20) NOT NULL AUTO_INCREMENT PRIMARY KEY,
81     data_collection_id BIGINT(20),
82     exam_id BIGINT(20),
83     attribute_json JSON,
84     step varchar(20) NOT NULL,
85     CONSTRAINT `fk_exam_dataCollectionType` FOREIGN KEY (`exam_id`)
    ↪ REFERENCES exam (`exam_id`) ON DELETE CASCADE,
86     CONSTRAINT `fk_dataCollectionType_exam` FOREIGN KEY
    ↪ (`data_collection_id`) REFERENCES data_collection_type
    ↪ (`data_collection_id`) ON DELETE CASCADE
87
88 ) ENGINE=InnoDB
89     CHARSET = utf8mb4
90     COLLATE = utf8mb4_unicode_ci;
91
92 CREATE TABLE IF NOT EXISTS data( /*création de la table data dans la DB*/
93     data_id BIGINT(20) NOT NULL AUTO_INCREMENT PRIMARY KEY,
94     data_key VARCHAR(90) NOT NULL UNIQUE,
95     data_value VARCHAR(255),
96     data_collection_id BIGINT(20) NOT NULL,
97
98     CONSTRAINT `fk_data_collectionType_data`
99     FOREIGN KEY (`data_collection_id`) REFERENCES
    ↪ data_collection_type(`data_collection_id`)
100 ) ENGINE=InnoDB
101     CHARSET = utf8mb4
102     COLLATE = utf8mb4_unicode_ci;
103
104 CREATE TABLE IF NOT EXISTS modification( /*création de la table modification
    ↪ dans la DB*/

```



```
105     modification_date DATE NOT NULL,  
106     user_id BIGINT(20),  
107     exam_id BIGINT(20),  
108  
109     PRIMARY KEY (user_id, exam_id),  
110     CONSTRAINT `FK_exam_modification_user` FOREIGN KEY (`exam_id`) REFERENCES  
111     ↪ exam (`exam_id`) ON DELETE CASCADE,  
112     CONSTRAINT `FK_user_modification_exam` FOREIGN KEY (`user_id`) REFERENCES  
113     ↪ user (`user_id`) ON DELETE CASCADE  
114 ) ENGINE=InnoDB  
115     CHARSET = utf8mb4  
116     COLLATE = utf8mb4_unicode_ci;
```

3. Visualisation de la modélisation de notre base de données avec le logiciel d'administration phpMyadmin

The screenshot shows the phpMyAdmin interface for a database named 'ClinicalChildData'. The left sidebar shows a tree view of the database structure, including tables like 'data', 'data_collection_type', 'exam', 'institution', 'modification', 'result', 'role', 'user', 'information_schema', 'mysql', and 'performance_schema'. The main panel displays a table listing the tables in the database. The table has columns: Table, Action, Lignes, Type, Interclassement, Taille, and Perte. The data is as follows:

Table	Action	Lignes	Type	Interclassement	Taille	Perte
data	Parcourir Structure Rechercher Insérer Vider Supprimer	30	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
data_collection_type	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
exam	Parcourir Structure Rechercher Insérer Vider Supprimer	50	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
institution	Parcourir Structure Rechercher Insérer Vider Supprimer	2	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
modification	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
result	Parcourir Structure Rechercher Insérer Vider Supprimer	2 345	InnoDB	utf8mb4_unicode_ci	1,7 Mio	-
role	Parcourir Structure Rechercher Insérer Vider Supprimer	4	InnoDB	utf8mb4_unicode_ci	32,0 kio	-
user	Parcourir Structure Rechercher Insérer Vider Supprimer	53	InnoDB	utf8mb4_unicode_ci	98,0 kio	-
8 tables	Somme	2 488	InnoDB	utf8mb4_general_ci	2,0 Mio	0 o

FIGURE 6.12 – Base de données modélisée

6.3 Importation des données du client vers la base de données

Nous avons reçu du client un document Excel contenant des données collectées sur différents nourrissons ayant subi une intervention chirurgicale. Afin d'importer ces données dans la base de données **ClinicalChildData** développée, nous avons procédé aux étapes suivantes :

- **Renommage des éléments de la première ligne du fichier en éléments compréhensibles par une base de données.**

Les noms contenus dans les cellules de la première ligne représentent l'attribut *data_key* de notre table Data. Cependant dans le fichier Excel, ces noms de cellules ont des espaces et des parenthèses, ce qui rend difficile la compréhension au niveau de la base de données. C'est la raison pour laquelle nous les avons renommé en concaténant² les mots avec le tiret de 8 (tiret bas). Pour éviter de modifier complètement les noms utilisés et fournis par le client, nous avons réalisé un tableau de concordance (ANNEXE C) entre les noms que nous reçus du fichier Excel et les noms que nous avons modifiés pour qu'ils soient compréhensibles par la base de données.

- **Utilisation de la librairie Apache POI pour la copie des éléments.**

Nous avons utilisé un code avec la programmation Java pour parcourir le fichier, copier les éléments et les stocker dans les tables adéquates. Nous avons fait usage de la librairie Apache POI qui est une librairie permettant aux programmeurs de créer, modifier et afficher des fichiers MS Office à l'aide de programmes Java. Il s'agit d'une bibliothèque open source développée et distribuée par Apache Software Foundation pour concevoir ou modifier des fichiers Microsoft Office à l'aide du programme Java. Cette librairie contient des classes et des méthodes pour décoder les données d'entrées de l'utilisateur ainsi qu'un fichier en documents MS Office [10]. Les valeurs des

2. Concatener : Action informatique qui consiste à relier deux chaînes de caractères en une seule

différentes cellules sont alors sauvegardées dans les tables adéquates.

Code Source : importation des données en java

```

1  /**
2   * @author: francine Youndzo
3   * class to import data from client excel file to the database
4   */
5  package ch.heigvd.clinicalChildDataCollection.importation;
6
7  import ch.heigvd.clinicalChildDataCollection.api.entity.*;
8  import
9      ↪ ch.heigvd.clinicalChildDataCollection.api.exception.ResourceNotFoundException;
10 import ch.heigvd.clinicalChildDataCollection.api.services.*;
11 import org.apache.poi.hssf.usermodel.HSSFWorkbook;
12 import org.apache.poi.ss.usermodel.Cell;
13 import org.apache.poi.ss.usermodel.DateUtil;
14 import org.apache.poi.ss.usermodel.Row;
15 import org.apache.poi.ss.usermodel.Sheet;
16 import org.springframework.beans.factory.annotation.Autowired;
17 import org.springframework.stereotype.Component;
18
19 import com.fasterxml.jackson.core.JsonGenerationException;
20 import com.fasterxml.jackson.databind.JsonMappingException;
21 import com.fasterxml.jackson.databind.ObjectMapper;
22
23 import java.io.File;
24 import java.io.FileInputStream;
25 import java.io.IOException;
26 import java.util.*;
27
28 @Component
29 public class ExcelReading {
30
31     private final DataService dataService;
32
33     private final DataCollectionTypeService dataCollectionTypeService;
34
35     private final ExamService examService;
36
37     private final ResultService resultService;
38
39     private final RoleService roleService;
40
41     private final UserService userService;
42
43
44     @Autowired
45     public ExcelReading(DataService dataService, DataCollectionTypeService
46         ↪ dataCollectionTypeService, ExamService examService, ResultService
47         ↪ resultService, RoleService roleService, UserService userService) {
48         this.dataService = dataService;
49         this.dataCollectionTypeService = dataCollectionTypeService;
50     }
51 }

```

```

48     this.examService = examService;
49     this.resultService = resultService;
50     this.roleService = roleService;
51     this.userService = userService;
52 }
53
54
55 public void startImportation() throws IOException {
56     // get file that needs to be mapped into object.
57     File file = new File("src/main/resources/HospitalDatabase.xls");
58     FileInputStream inputStream = new FileInputStream(file);
59     List<String> step = Arrays.asList("start", "induction", "traumatic",
60     ↪ "after_operation", "24_after_operation");
61     List<String> roles = Arrays.asList("admin", "nurse", "doctor",
62     ↪ "patient");
63     // create Rôle
64     roles.stream().forEach(e ->
65     ↪ creatorRole(e));
66
67     // create super admin
68     creatorAdmin();
69
70     // get workbook and sheet
71     HSSFWorkbook workbook = new HSSFWorkbook(inputStream);
72     Sheet sheet = workbook.getSheetAt(0);
73
74     List<DataCollectionType> dataCollectionTypes =
75     ↪ Arrays.asList(creatDataCollectionType("childData"),
76     ↪ creatDataCollectionType("operationData"),
77     ↪ creatDataCollectionType("motherData"),
78     ↪ creatDataCollectionType("motherData"),
79     ↪ creatDataCollectionType("dataMonitor"));
80
81     int index = 0;
82
83     List<String> keys = new ArrayList<>();
84
85     Iterator<Row> iterator = sheet.iterator();
86
87     while (iterator.hasNext() && index < 52) {
88         ++index;
89         Row currentRow = iterator.next();
90
91         if (currentRow.getRowNum() == 0) {
92             for (int i = 0; i < 102; ++i) {
93                 if (i <= 10) {
94                     keys.add(currentRow.getCell(i).getStringCellValue()
95                     ↪ .split("_")[0]);
96                     creatData(currentRow.getCell(i).getStringCellValue()
97                     ↪ .split("_")[0], dataCollectionTypes.get(0));
98                 } else {
99                     keys.add(currentRow.getCell(i).getStringCellValue()

```

```

96         .split("_")[0]);
97         creatData(currentRow.getCell(i).getStringCellValue().
98             split("_")[0], dataCollectionTypes.get(1));
99     }
100 }
101     continue;
102 }
103
104     User user =
105         ↪ createUser(currentRow.getCell(2).getStringCellValue());
106     String anesthesiaType = " ";
107     if(currentRow.getCell(8)!=null) {
108         anesthesiaType = (currentRow.getCell(8).getStringCellValue());
109     }
110     Exam exam = createExam(user, anesthesiaType);
111     Map<String, Object> resultChilddata = new HashMap<String,
112         ↪ Object>();
113     Map<String, Object> resultOperationData = new HashMap<String,
114         ↪ Object>();
115     for (int i = 0; i < 17; ++i) {
116
117         if (currentRow.getCell(i) != null) {
118             if (i <= 11) {
119                 resultChilddata.put(keys.get(i),
120                     ↪ getValue(currentRow.getCell(i)));
121             } else {
122                 for(int j = i; j < 102; ) {
123                     resultOperationData.put((sheet.getRow(0).getCell(j).
124                         getStringCellValue()),
125                         ↪ getValue(currentRow.getCell(j)));
126                     j+=5;
127                 }
128                 createResult(resultChilddata, resultOperationData,
129                     ↪ step.get((i -2)%5), exam, dataCollectionTypes);
130             }
131         }
132     }
133 }
134     workbook.close();
135 }
136
137 private DataCollectionType creatDatacollectionType(String
138     ↪ dataCollectionTypeName) {
139     return dataCollectionTypeService.findByName(dataCollectionTypeName)
140         .orElseGet(
141             () -> dataCollectionTypeService.create(new
142                 ↪ DataCollectionType(dataCollectionTypeName))
143         );
144 }
145
146 private User createUser(String username) {

```

```

143     return userService.findBySurname(username).orElseGet(
144         () -> userService.create(new User(username,
145             roleService.findByRoleName("patient").
146                 orElseThrow(() -> new
147                     ↪ ResourceNotFoundException(Role.class,
148                     ↪ "patient"))));
147     );
148 }
149
150 private Exam createExam(final User user, String anesthesiaType) {
151     Exam exam = new Exam(user, anesthesiaType);
152     return examService.findById(user.getUserId()).orElseGet(
153         () -> examService.create(exam)
154     );
155 }
156
157 /**
158  * this methode persist on the key and link it to datacollectionType
159  * @param dataKeys
160  * @param dataCollectionType
161  * @return
162  */
163 private Data creatData(final String dataKeys, DataCollectionType
164     ↪ dataCollectionType) {
165     Data data = new Data();
166     data.setDataCollectionType(dataCollectionType);
167     data.setData_key(dataKeys);
168     return dataService.findByDataKey(dataKeys).orElseGet(
169         () -> dataService.create(data)
170     );
171 }
172
173 /**
174  * this method get the value of a cell depended of the type of the data
175  * @param cell
176  * @return
177  */
178 private String getValue(Cell cell) {
179     switch (cell.getCellTypeEnum()) {
180         case BOOLEAN:
181             return String.valueOf(cell.getBooleanCellValue());
182
183         case STRING:
184             return cell.getRichStringCellValue().getString();
185
186         case NUMERIC:
187             if (DateUtil.isCellDateFormatted(cell)) {
188                 return String.valueOf(cell.getDateCellValue());
189             } else {
190                 return String.valueOf(cell.getNumericCellValue());
191             }
192     }
193 }

```

```

194         case FORMULA:
195             return String.valueOf(cell.getCellFormula());
196     }
197     return " ";
198 }
199 }
200
201 private void createResult(final Map<String, Object> childData, final
↪ Map<String, Object> operationData,
202                          String step, final Exam exam, final
↪ List<DataCollectionType> dataCollectionType)
↪ {
203     List<Result> result = resultService.findByExamAndStep(exam,
↪ dataCollectionType.get(0));
204     if(result.isEmpty()) {
205         resultService.create( new Result(exam, dataCollectionType.get(0),
↪ childData, step));
206     }
207     resultService.create(new Result(exam, dataCollectionType.get(1),
↪ operationData, step));
208     operationData.clear();
209 }
210
211 private Role creatorRole(final String roleName) {
212     return roleService.findByName(roleName).orElseGet( ()->
213         roleService.create(new Role(roleName))
214     );
215 }
216
217 /**
218  * this method help to authomatically create and admin when the data is
↪ import into the database
219  * @return
220  */
221 private User creatorAdmin() {
222     User admin = new User();
223     admin.setEmail("admin@.ch");
224     admin.setSurname("admin");
225     admin.setPassword("admin");
226     admin.setRole(roleService.findByName("admin").orElseThrow( ()->
227         new ResourceNotFoundException(Role.class, "admin")
228     ));
229     Optional<User> userSave = userService.findByName("admin");
230     if (userSave.isPresent()) {
231         return userSave.get();
232     }
233     return userService.create(admin);
234 }
235 }

```

NB : Les données sont stockées en fonction du type de collection. L'on s'assure que la valeur d'une cellule se retrouve dans la bonne collection. Le fichier Excel comprend des valeurs chirurgicales enregistrées pendant les cinq moments de la collecte (au début de l'opération, pendant l'opération, pendant le moment d'induction, pendant

le moment traumatique et 24 heures après l'opération). La table **result** contient le résultat de l'importation des données médicales et la table **user** contient les informations concernant l'utilisateur qui a pour rôle patient. Les valeurs de la cellule **Surname** seront insérées dans la table **user** (`user.surname`), mais pas dans la cellule **result** comme les autres données du fichier Excel. Néanmoins, pour retrouver un patient dont les données sont stockées dans la table **result**, on utilisera l'identifiant de la table **exam** (`exam_id`) qui nous permettra d'obtenir l'identifiant du patient dans la table **user** (`user_id`) puisque ces deux tables sont liées. Ceci grâce au mappage entre la table **exam** et la table **user** avec la table **result**.

Résultat de l'importation des données médicales : Visualisation avec php-Myadmin.

Serveur: db » Base de données: ClinicalChildData » Table: result

[Parcourir](#) [Structure](#) [SQL](#) [Rechercher](#) [Insérer](#) [Exporter](#) [Importer](#) [Privileges](#) [Opérations](#)

Affichage des lignes 0 - 24 (total de 2855, traitement en 0.0044 seconde(s).)

```
SELECT * FROM `result`
```

☐ Profilage [Éditer en ligne] [Éditer] [Expliquer SQL]

1 > >> | Nombre de lignes : 25 Filtre les lignes: Chercher dans cette table Trier par clé : Aucun(e)

Options		result_id	data_collection_id	exam_id	attribute_json	step
<input type="checkbox"/>	Éditer Copier Supprimer	1	1	1	("ID ":"Neo_2596","Chirurgical":"Tue Sep 19 00:00:..."	start
<input type="checkbox"/>	Éditer Copier Supprimer	2	2	1	("BR_start":"50.0","Systolic_Blood_Pressure_start"...	start
<input type="checkbox"/>	Éditer Copier Supprimer	3	2	1	("Diastolic_blood_pressure_induction":"53.0","BE_i...	induction
<input type="checkbox"/>	Éditer Copier Supprimer	4	2	1	("HR_traumatic":"160.0","Diastolic_blood_pressure_...	traumatic
<input type="checkbox"/>	Éditer Copier Supprimer	5	2	1	("Na_after_operation":"140.0","BR_after_operation"...	after_operation
<input type="checkbox"/>	Éditer Copier Supprimer	6	2	1	("BR_24h_after_operation":"38.0","Diastolic_blood_...	24_after_operation
<input type="checkbox"/>	Éditer Copier Supprimer	7	1	2	("ID ":"Neo_2535","Chirurgical":"Mon Sep 25 00:00:..."	start
<input type="checkbox"/>	Éditer Copier Supprimer	8	2	2	("BR_start":"38.0","Systolic_Blood_Pressure_start"...	start
<input type="checkbox"/>	Éditer Copier Supprimer	9	2	2	("Diastolic_blood_pressure_induction":"65.0","BE_i...	induction
<input type="checkbox"/>	Éditer Copier Supprimer	10	2	2	("HR_traumatic":"163.0","Diastolic_blood_pressure_...	traumatic
<input type="checkbox"/>	Éditer Copier Supprimer	11	2	2	("Na_after_operation":"138.4","BR_after_operation"...	after_operation
<input type="checkbox"/>	Éditer Copier Supprimer	12	2	2	("BR_24h_after_operation":"40.0","Diastolic_blood_...	24_after_operation
<input type="checkbox"/>	Éditer Copier Supprimer	13	1	3	("ID ":"Neo_2756","Chirurgical":"Fri Jan 13 00:00:..."	start
<input type="checkbox"/>	Éditer Copier Supprimer	14	2	3	("BR_start":"35.0","Systolic_Blood_Pressure_start"...	start
<input type="checkbox"/>	Éditer Copier Supprimer	15	2	3	("Diastolic_blood_pressure_induction":"38.0","BE_i...	induction

FIGURE 6.13 – Résultat de l'importation des données

6.4 Exportation des données de la base de données sous forme de documents Excel

Pour exporter les données afin de préparer le rapport annuel du département chirurgie, nous avons écrit un code de programmation en java qui grâce à la librairie Apache POI parcourt les tables de la base de données, sélectionne les colonnes à exporter ainsi que leurs valeurs, puis les stocke dans un document Excel. Le programme utilise JPA (Java Persistence API) qui permet de travailler directement avec des objets plutôt qu'avec des instructions SQL. JPA permet de récupérer les informations depuis la base de données sous forme d'objet Java, les déséréalise à l'aide de Apache POI puis écrit ses propriétés dans le fichier Excel. Pour le test nous avons choisi d'exporter les données suivantes : **surname, case history number, gender, age, weight et diagnosis**

1	2	3	4	5	6
Case history number	Surname	Gender	Age	Weight	Diagnosis

FIGURE 6.14 – Données à exporter

6.5 Transformation des données sous le format CDM

La transformation des données sous le format OMOP Common Data Format facilite les études sur les données reçues à l'OHDSI. Les données sont issues des bases de données différentes avec des langues différentes. Ce format permet aux institutions de normaliser leurs données avant de les soumettre à l'organisme spécialisé à des fins d'étude et de statistiques. Notre travail s'effectuant dans le domaine pédiatrique, nous avons choisi d'utiliser le modèle de données commun (PEDSnet CDM) spécifique à la pédiatrie pour le stockage des données. L'utilisation du format PEDSnet CDM permet d'ajouter rapidement des domaines de données ou des éléments de données nécessaires aux enquêteurs pédiatriques. Ce format est basé sur l'OMOP CDM et se concentre fortement sur la normalisation de la terminologie.

6.5.1 Processus

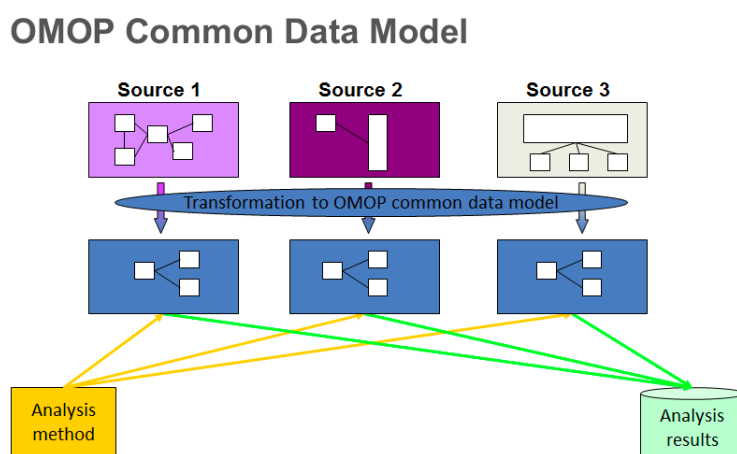


FIGURE 6.15 – Processus de transformation [7]

La figure ci-dessus présente le processus de transformation des données en CDM. Les bases de données sont des sources et proviennent de différentes institutions du monde qui utilisent différents gestionnaires de base de données comme MySQL, PostgreSQL ou MariaDB dans notre cas. Après avoir développé la base de données, l'on la transforme en OMOP CDM et l'OHSDI s'occupe de faire des analyses avec les techniques d'analyse de données (Data Analyst/ Data Science).

6.5.2 Transformation

Transformer les données de notre base de données en CDM n'est pas chose aisée et peut faire l'objet d'un projet de recherche. Pour transformer les données nous devons impérativement respecter le vocabulaire du CDM. Un aperçu des tables de l'OMOP CDM version 6.0 (Les

versions évoluent rapidement) est le suivant :

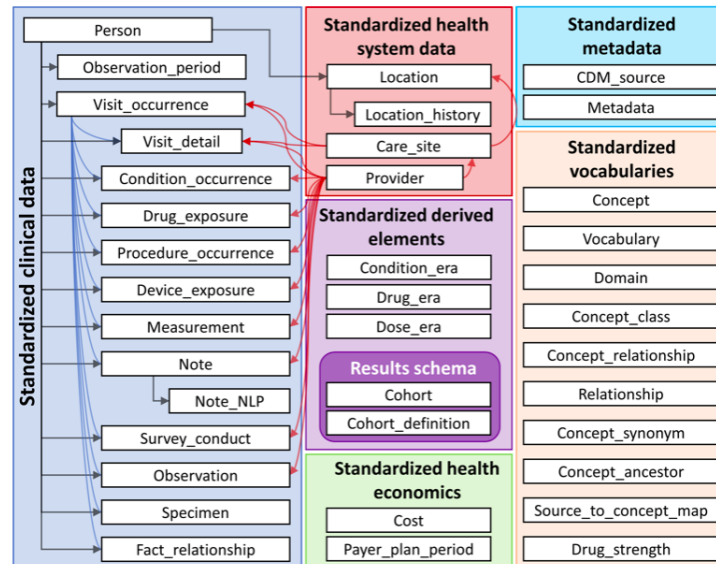


FIGURE 6.16 – Tables dans de OMOP CDM

[7]

Selon le code de couleur de la figure ci-dessus, la partie bleue identifie la table représentative d'un patient (PERSON) et toutes ses informations. Dans notre base de données le patient est identifié dans la table **user** nous allons donc transformer cette table sous le format CDM. La table PERSON du CDM contient 20 colonnes, mais toutes ne sont pas nécessaires. Nous avons uniquement mapper les colonnes qui nous intéresse comme le montre la figure ci-dessous.

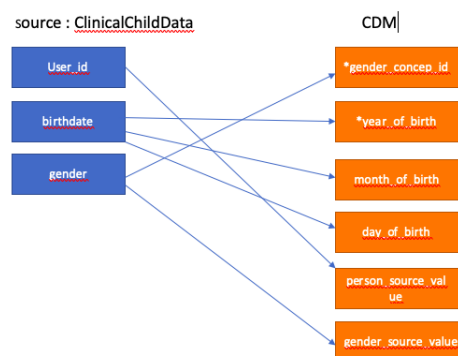


FIGURE 6.17 – Mapping des colonnes

ce qui revient à avoir la transformation suivante :

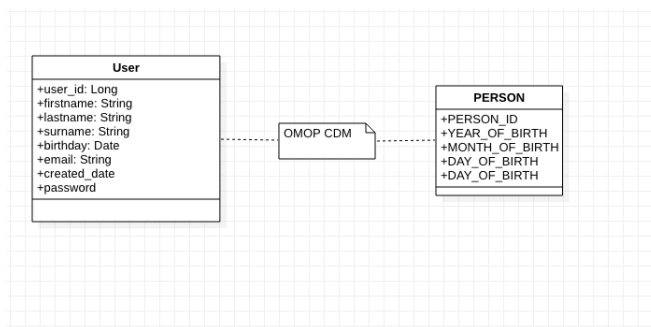


FIGURE 6.18 – CDM transformation

Les informations relatives à un examen sont stockées dans la table **result** de notre base de données. Pour les transformer en CDM, il faudra la correspondre avec la table **Measurement** de l'OMOP CDM qui est la suivante :

Measurement
+measurement_id
+person_id
+measurement_concept_id
+measurement_date
+measurement_datetime
+measurement_time
+measurement_type_concept_id
+operator_concept_id
+value_as_concept_id
+unit_concept_id
+range_low
+range_high
+provider_id
+visit_occurrence_id
+visit_detail_id
+measurement_source_value
+unit_source_value
+measurement_source_concept_id
+value_source_value

FIGURE 6.19 – Table Measurement

Remarque : Au vu du temps à disposition pour le déroulement de ce travail nous n'avons pas eu la possibilité de mapper toutes nos tables sous le format CDM. Ainsi, nous avons pu implémenter et tester la faisabilité technique du mapping qui pourrait être complété par la suite pour les tables restants de la même manière.

Chapitre 7

Implémentation

Ce chapitre comprend les étapes d'implémentation de l'application. L'architecture étant structurée en trois parties, il sera question de les faire interagir entre elles en liant les différents niveaux de l'architecture **tree-tiers**.

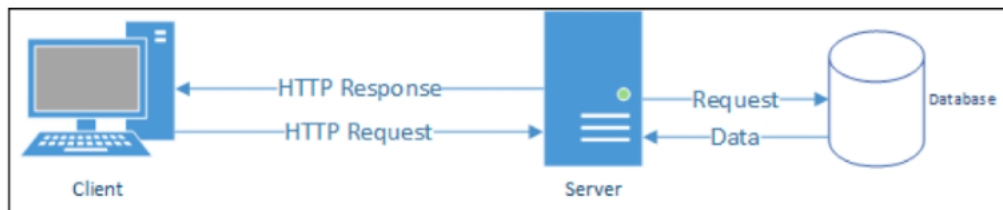


FIGURE 7.1 – Interaction dans l'architecture

7.1 Implémentation du backend

Comme mentionné plus haut (Voir Chapitre 3), nous avons décidé d'utiliser le framework Spring Boot pour implémenter notre backend. Spring Boot est ainsi installé sur l'IDE (Integrated Development Environment) **IntelliJ IDEA** et tourne avec la version de **java 11.0.2**. Grâce au DTO du backend, l'application pourra communiquer avec la base de données en transformant les attributs des tables en objet Java. Cette partie représente le serveur de l'application. La figure ci-dessous présente la structure du backend.

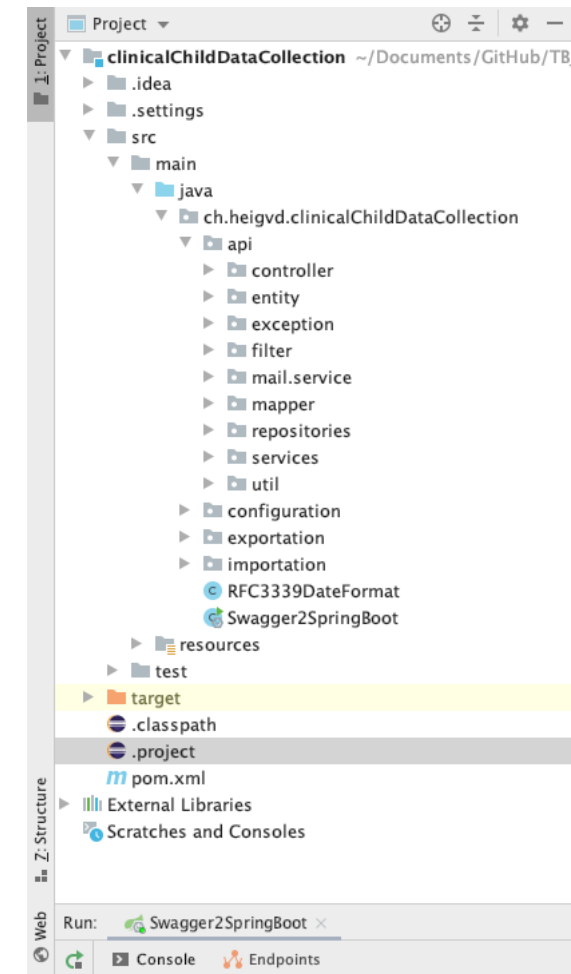


FIGURE 7.2 – Structure du backend

La structure du projet au niveau backend est divisé en plusieurs parties :

- Les entités
- Les contrôleurs
- Les services
- Les référentiels
- Les mappers

7.1.1 Entity

Une "entity" représente une table stockée dans une base de données. Chaque instance d'une entité représente une ligne dans la table. Pour chaque table de notre base de données nous aurons une entité associée, comme le montre l'arborescence suivante.

```

air-de-youndzo:entity youndzofrancine$ tree
.
├── Data.java
├── DataCollectionType.java
├── DataTranslatableEntity.java
├── Exam.java
├── Institution.java
├── JsonType.java
├── Modification.java
├── ModificationPk.java
├── Result.java
├── Role.java
└── User.java

```

FIGURE 7.3 – Entities

7.1.2 Controller

Un contrôleur contrôle le flux des données et fait des mises à jour aux services chaque fois que les données changent. Son rôle est de récupérer les données en fonction de la requête, de les filtrer, de les vérifier, de choisir le traitement approprié, et finalement d'envoyer la réponse au client. L'utilisation des contrôleurs a pour effet de donner une structure hiérarchique à l'application, ce qui facilite la compréhension du code et l'accès rapide aux parties à modifier. Chaque fois que l'utilisateur va cliquer sur un lien ou chaque fois qu'il va soumettre un formulaire, une requête HTTP arrivera au contrôleur. A ce moment, le contrôleur décide ce qu'il faut faire et gère l'organisation. L'arborescence des différents contrôleurs de notre API est la suivante :

```

air-de-youndzo:controller youndzofrancine$ tree
.
├── AuthenticateController.java
├── ExamController.java
├── RegisterApiController.java
├── ResetPasswordController.java
├── UpdatePasswordController.java
└── UsersApiController.java

```

FIGURE 7.4 – Controllers

7.1.3 Services

Un service est une classe comprenant des méthodes permettant de traiter une requête. Après avoir reçu la requête, identifie le service adéquat pour la traiter. Par exemple, si l'utilisateur veut obtenir la liste des patients, le contrôleur identifie le service qui contient la méthode pouvant fournir ces données et fait un appel à ce dernier. L'arborescence des services de notre API est la suivante :

```
lair-de-youndzo:services youndzofrancine$ tree
.
├── AbstractCrudService.java
├── CrudService.java
├── DataCollectionTypeService.java
├── DataService.java
├── ExamService.java
├── InstitutionService.java
├── ModificationService.java
├── ResultService.java
├── RoleService.java
└── UserService.java
```

FIGURE 7.5 – Services

7.1.4 Repository

Un référentiel (repository) est une interface que JPA met à disposition avec des méthodes déjà implémentées pour facilement effectuer les requêtes à la base de données. Il fournit des méthodes de recherche tels que : `findById()`, `findByName` ou `findByNameAndRole()`. Il encapsule la logique requise pour accéder aux sources de données. L'arborescence de nos référentiels est la suivante :

```
lair-de-youndzo:repositories youndzofrancine$ tree
.
├── DataCollectionTypeRepository.java
├── DataRepository.java
├── ExamRepository.java
├── InstitutionRepository.java
├── ModificationRepository.java
├── ResultRepository.java
├── RoleRepository.java
└── UserRepository.java
```

FIGURE 7.6 – Repository

7.1.5 Mapper

Le mapper permet de transformer les objets DTO (Data Transfer Object) qui sont les données contenues dans les requêtes du client, en objets Java (entités) afin de les stocker dans la base de données.

```
lair-de-youndzo:mapper youndzofrancine$ tree
.
├── ExamMapper.java
└── UserMapper.java
```

FIGURE 7.7 – Mapper

7.1.6 Sécurité

La sécurité vise à protéger l'accès aux données des patients qui composent leurs dossiers médicaux. Un exemple de menace pourrait être une personne non autorisée qui accède à des données vitales d'un patient et modifie des valeurs ou les supprime. Afin d'éviter toute intrusion dans le système, nous allons utiliser des fonctionnalités du JWT. Le JWT (JSON Web Token) est une norme ouverte (RFC 7519) qui définit un moyen compact

et autonome de transmettre en toute sécurité des informations entre les parties en tant qu'objet JSON. Ces informations peuvent être vérifiées et rendues fiables car elles sont signées numériquement[2]. Un exemple de scénario d'utilisation du JWT dans notre API est la suivante :

- Utilisation du endpoint ¹ /login pour s'authentifier en tant qu'utilisateur.
- Vérification dans la base de données si l'utilisateur existe et si ses credentials (email et mot de passe) sont justes.
- Une fois que les credentials sont correctes, on recevra un code de statut positif (200OK) et un jeton. JWT standardise la génération et la signature du jeton.
- Une fois qu'on a le jeton, on peut le passer dans des requêtes pour donner aux utilisateurs authentifiés le droit d'accès à certaines fonctionnalités telles que l'enregistrement de nouveaux patients ou la modification des données d'un patient.

Code Source : Génération du token avec JWT

```

1 package ch.heigvd.clinicalChildDataCollection.api.util;
2
3 import ch.heigvd.clinicalChildDataCollection.api.entity.User;
4 import com.auth0.jwt.JWTVerifier;
5 import com.auth0.jwt.exceptions.JWTCreationException;
6 import io.jsonwebtoken.Claims;
7 import io.jsonwebtoken.JwtException;
8 import io.jsonwebtoken.Jwts;
9 import io.jsonwebtoken.SignatureAlgorithm;
10 import org.springframework.beans.factory.annotation.Value;
11 import org.springframework.stereotype.Component;
12
13 import java.util.Date;
14
15 @Component
16 public class JWTutils {
17
18     @Value("${SCHEMA}")
19     private static String SCHEMA = "Bearer";
20     @Value("${ISSUER}")
21     private static String ISSUER;
22     private static String SECRET_KEY = "francine";
23     private static long validity=24*60*60*1000;
24     private static JWTVerifier verifier;
25
26     public static String generateToken (User userentity){
27
28         Date now = new Date();
29         System.out.println(SECRET_KEY);
30         Date val = new Date(now.getTime() + validity);
31         Claims claims = Jwts.claims().setSubject(userentity.getEmail());
32         claims.put("role", userentity.getRole().getRoleName());
33         try {
34             return Jwts.builder()

```

1. endpoint : point d'entrée dans un canal de communication lorsque deux systèmes interagissent

```

35         .setClaims(claims)
36         .setIssuedAt(now)
37         .setExpiration(val)
38         .signWith(SignatureAlgorithm.HS256, SECRET_KEY)
39         .compact();
40     } catch (JWTCreationException e){
41         e.printStackTrace();
42         return null;
43     }
44 }
45
46 public static String extractToken(String header){
47     if (header == null || header.length() < SCHEMA.length() + 1) {
48         return null;
49     }
50
51     return header.substring(SCHEMA.length() + 1);
52 }
53 public static boolean verifyToken(String token) {
54     try {
55         Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token);
56         return true;
57     } catch (JwtException | IllegalArgumentException ignored) {
58     }
59     return false;
60 }
61
62 public static String getEmail(String token){
63     String str = "role";
64     return Jwts.parser().setSigningKey(SECRET_KEY)
65         .parseClaimsJws(token).getBody().getSubject();
66 }
67
68 public static String getRole(String token){
69
70     return (String) Jwts.parser().setSigningKey(SECRET_KEY)
71         .parseClaimsJws(token).getBody().get("role");
72 }
73
74 public static String getStatut(String token){
75     return (String) Jwts.parser().setSigningKey(SECRET_KEY)
76         .parseClaimsJws(token).getBody().get("statut");
77 }
78
79 }

```

Code Source : Sécurisation

```

1 package ch.heigvd.clinicalChildDataCollection.api.controller;
2 import ch.heigvd.clinicalChildDataCollection.api.LoginApi;
3 import ch.heigvd.clinicalChildDataCollection.api.entity.User;
4 import
    ↳ ch.heigvd.clinicalChildDataCollection.api.exception.AuthenticationFailedException;

```

```

5 import
  ↳ ch.heigvd.clinicalChildDataCollection.api.exception.ResourceNotFoundException;
6 import ch.heigvd.clinicalChildDataCollection.api.model.Credentials;
7 import ch.heigvd.clinicalChildDataCollection.api.services.UserService;
8 import ch.heigvd.clinicalChildDataCollection.api.util.ErrorDescription;
9 import ch.heigvd.clinicalChildDataCollection.api.util.JWTutils;
10 import ch.heigvd.clinicalChildDataCollection.api.util.PasswordUtile;
11 import com.fasterxml.jackson.databind.ObjectMapper;
12 import org.springframework.beans.factory.annotation.Autowired;
13 import org.springframework.http.ResponseEntity;
14 import org.springframework.web.bind.annotation.CrossOrigin;
15 import org.springframework.web.bind.annotation.RequestBody;
16 import org.springframework.web.bind.annotation.RestController;
17
18 import javax.servlet.http.HttpServletRequest;
19 import javax.servlet.http.HttpServletResponse;
20 import javax.validation.Valid;
21 import java.io.IOException;
22 import java.util.Optional;
23
24 @RestController
25 @CrossOrigin(origins = "*")
26 public class AuthenticateController implements LoginApi {
27
28     private UserService userService;
29
30     final HttpServletRequest httpServletRequest;
31
32     @Autowired
33     public AuthenticateController(UserService userService, HttpServletRequest
  ↳ httpServletRequest) {
34         this.userService = userService;
35         this.httpServletRequest = httpServletRequest;
36     }
37
38     public ResponseEntity<Void> login(@Valid @RequestBody Credentials credentials)
  ↳ {
39         User user = userService.findByEmail(credentials.getEmail()).orElseThrow(()->
40         new ResourceNotFoundException(User.class, credentials.getEmail()));
41
42         if(!PasswordUtile.isPasswordValid(credentials.getPassword(),
  ↳ user.getPassword(), user.getSalt())){
43             throw new AuthenticationFailedException();
44         }
45
46         return ResponseEntity
47             .ok()
48             .header("Authorization", JWTutils.generateToken(user))
49             .build();
50     }
51     private void sendError(HttpServletResponse response, int status, String
  ↳ errorCode, String message) throws IOException {
52         response.setStatus(status);

```

```
53     response.setHeader("Content-Type", "application/json");
54
55     ObjectMapper mapper = new ObjectMapper();
56     response.getWriter().write(mapper.writeValueAsString(new
57         ↵ ErrorDescription(errorCode, message)));
58 }
```

7.1.7 Configuration

Diverses propriétés peuvent être spécifiées dans le fichier **application.properties**. Ce fichier contient toutes les informations relatives au lancement de l'application et à la connexion avec la base de données. Ci-dessous quelques propriétés de l'API :

Code Source : Properties

```
1 springfox.documentation.swagger.v2.path=/api-docs
2 server.servlet.context-path=/api/clinicalChildData
3 server.port=8087
4 spring.jackson.date-format=ch.heigvd.clinicalChildDataCollection.RFC3339DateFormat
5 spring.jackson.serialization.WRITE_DATES_AS_TIMESTAMPS=false
6
7 logging.level.org.hibernate.SQL=DEBUG
8 logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
9
10 spring.jpa.hibernate.ddl-auto=validate
11 spring.datasource.initialization-mode=always
12 hibernate.dialect=org.hibernate.dialect.MariaDBDialect
13 spring.jpa.database-platform=org.hibernate.dialect.MariaDBDialect
14 spring.datasource.url=jdbc:mariadb://localhost:3307/ClinicalChildData
15 spring.datasource.username=root
16 spring.datasource.password=adminpw
17 spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
```

Les règles de sécurité sont les suivantes :

1. Un utilisateur avec le rôle **admin** est autorisé à enregistrer un utilisateur avec le rôle **doctor** uniquement.
2. Un utilisateur avec le rôle **doctor** est autorisé à enregistrer un utilisateur avec le rôle **nurse** et le rôle **patient** uniquement.
3. Un utilisateur avec le rôle **doctor** est autorisé à faire un examen, enregistrer les données d'un patient, modifier les paramètres de celui-ci et afficher les logs des modifications effectuées sur les données.
4. Un utilisateur avec le rôle **nurse** n'est pas autorisé à créer un patient et à créer un examen.
5. Un utilisateur avec le rôle **nurse** est autorisé à modifier les paramètres d'un examen déjà créés par l'utilisateur avec le rôle **doctor**.

7.2 Implémentation du frontend

L'implémentation de cette partie permet à l'utilisateur final d'utiliser l'interface graphique. Pour cela, nous avons choisi d'utiliser le framework d'Angular. Ce dernier récupère les données du back-end, les affiche à l'utilisateur via une interface contenant des boutons ou des formulaires. Ces boutons et ces formulaires permettent de déclencher des modifications côté back-end à partir des requêtes. On pourrait donc dire qu'Angular fait l'intermédiaire entre l'utilisateur et le back-end. La structure de notre projet Angular est la suivante

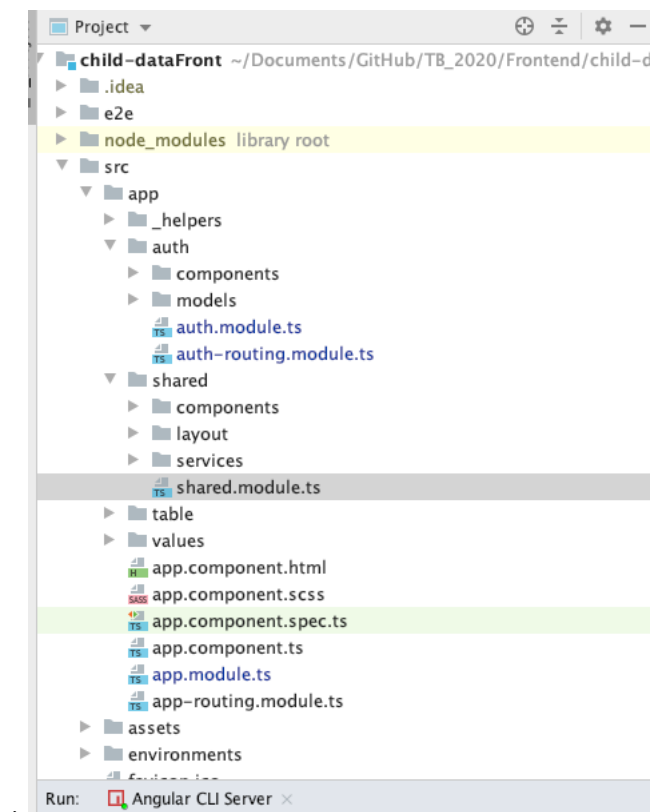


FIGURE 7.8 – Interaction dans l'architecture

L'architecture d'une application Angular repose sur certains concepts fondamentaux tels que l'existence dans le projet d'au moins un module et un composant. Les composants définissent des vues (page html). Ils communiquent également avec d'autres composants et services pour apporter des fonctionnalités à l'application. Les modules quant à eux comprennent un ou plusieurs composants. Ils ne contrôlent aucune page html. Ils déclarent quels composants peuvent être utilisés et quelles classes seront injectées par l'injecteur de dépendances.

Nous avons dans notre projet deux modules et huit composants. Pour se connecter à notre projet Angular il suffira d'entrer dans l'URL <http://localhost:4200> dans le navigateur web pour avoir accès à l'interface utilisateur. Les différentes pages implémentées sont les

suivantes :

- Page de login

The screenshot displays the login interface for 'Clinical Child Data'. At the top, a light gray header contains the application name and links for 'Register', 'Login', and 'Reset-Password'. Below this is a blue status bar with a red rocket icon and the text 'child-dataFront app is running!'. The main content area is white and contains a 'LOGIN' section. This section includes an 'Email' label and input field, a 'Password' label and input field, and a blue 'Login' button. A small copyright symbol is visible at the bottom of the login form.

FIGURE 7.9 – Login page

- Page d'enregistrement des utilisateur

Clinical Child Data Register Login Reset-Password

child-dataFront app is running!

Register New person

Email
Enter email

Firstname
Enter firstname

Lastname
Enter lastname

Surname
Enter surname

Select a role
Doctor
Nurse
Patient

Password
Password

Register

FIGURE 7.10 – Register page

- Page de réinitialisation du mot de passe

Clinical Child Data Register Login Reset-Password

child-dataFront app is running!

Reset Password

Account Email
Enter email

Reset Password ↗

FIGURE 7.11 – Reset-password page

- Page pour effectuer un examen

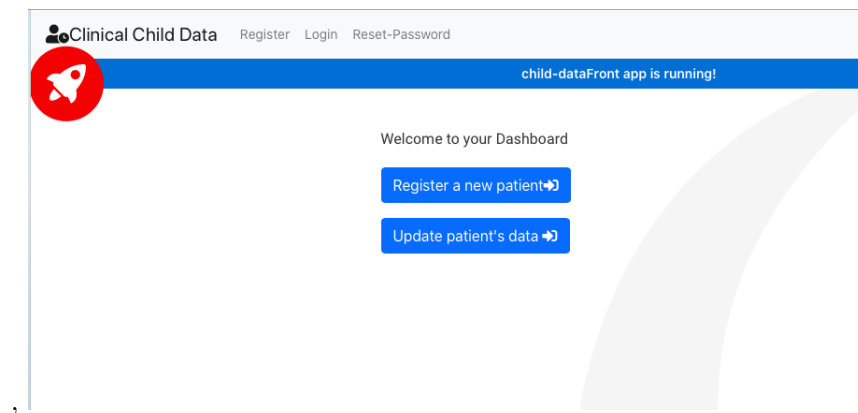


FIGURE 7.12 – Create an exam page

- Page de sélection du type de données à collecter

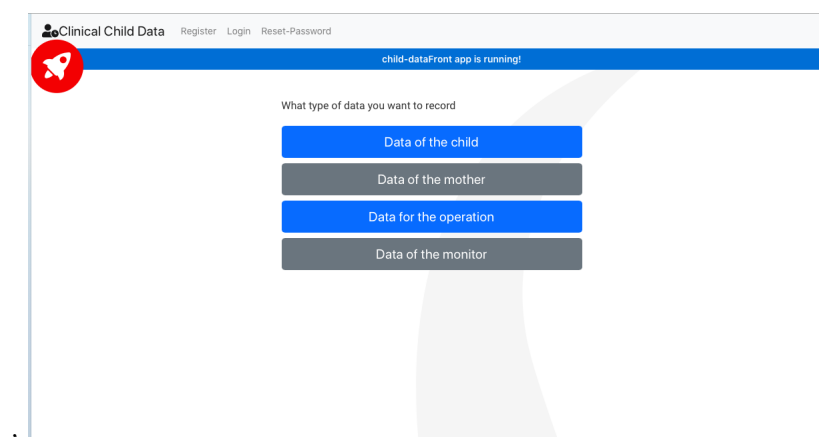


FIGURE 7.13 – Data-type page

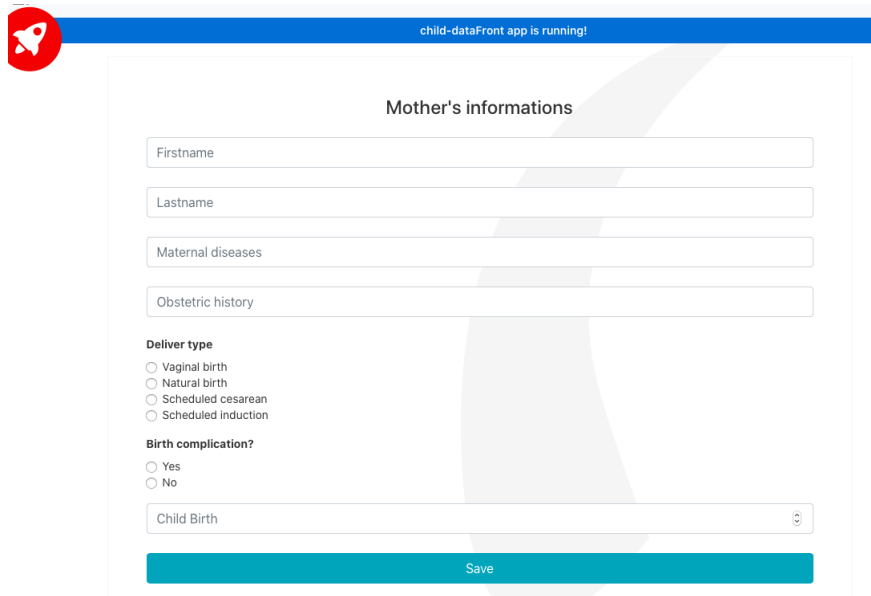
- Page de sélection du moment de la collecte des données de l'opération

FIGURE 7.14 – Five points of collect page

- Formulaire d'enregistrement des données du nourrisson

FIGURE 7.15 – Child data form

• Formulaire d'enregistrement des données de la mère



child-dataFront app is running!

Mother's informations

Firstname

Lastname

Maternal diseases

Obstetric history

Deliver type

☐ Vaginal birth

☐ Natural birth

☐ Scheduled cesarean

☐ Scheduled induction

Birth complication?

☐ Yes

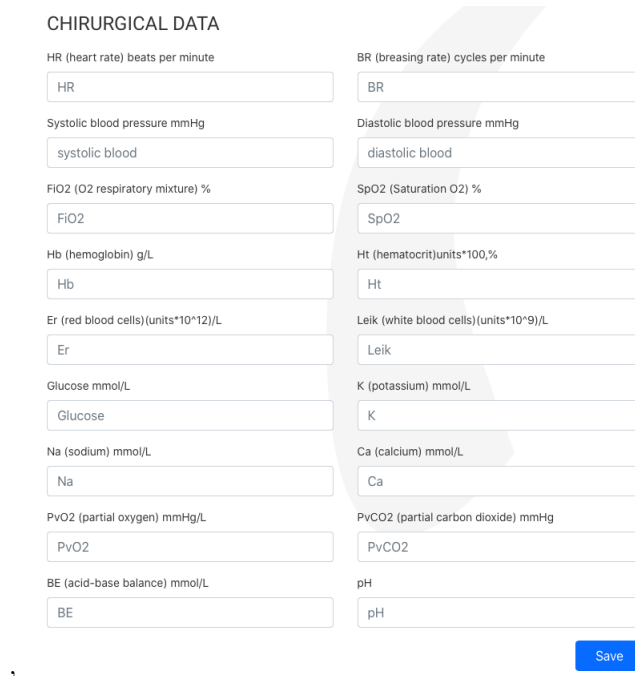
☐ No

Child Birth

Save

FIGURE 7.16 – Mother data form

• Formulaire d'enregistrement des données de l'opération



CHIRURGICAL DATA

HR (heart rate) beats per minute

HR

Systolic blood pressure mmHg

systolic blood

FIO2 (O2 respiratory mixture) %

FIO2

Hb (hemoglobin) g/L

Hb

Er (red blood cells)(units*10¹²)/L

Er

Glucose mmol/L

Glucose

Na (sodium) mmol/L

Na

PvO2 (partial oxygen) mmHg/L

PvO2

BE (acid-base balance) mmol/L

BE

BR (breasing rate) cycles per minute

BR

Diastolic blood pressure mmHg

diastolic blood

SpO2 (Saturation O2) %

SpO2

Ht (hematocrit)units*100,%

Ht

Leik (white blood cells)(units*10⁹)/L

Leik

K (potassium) mmol/L

K

Ca (calcium) mmol/L

Ca

PvCO2 (partial carbon dioxide) mmHg

PvCO2

pH

pH

Save

FIGURE 7.17 – Operation data form

- Page de choix du fichier d'enregistrements du moniteur

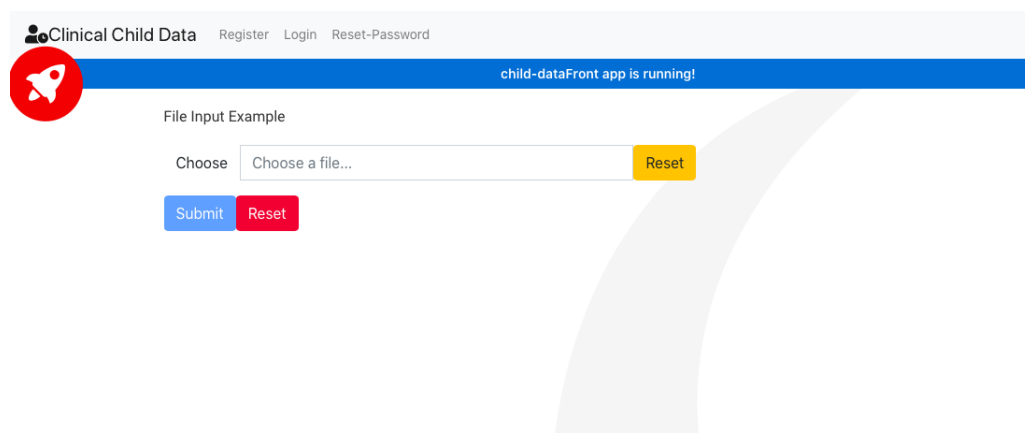


FIGURE 7.18 – Monitor data page

Chapitre 8

Tests et résultats

Dans ce chapitre, nous allons effectuer plusieurs tests permettant ainsi de montrer que notre base de données est opérationnelle. Le chapitre comportera des scénarios de test qui seront démontrer sous formes de videos fournis en annexe de ce document.

8.1 Importation des données depuis le fichier Excel du client (cf ANNEXE video_importation)

Pour importer les données nous avons programmé en java un code qui se lance au démarrage de l'application. Ce code effectue une importation automatique des données de façon dynamique depuis le fichier Excel du client. **Démonstration de l'importation des données**

8.2 Sécurité des données (cf ANNEXE video_sécurité)

Pour pouvoir tester la sécurité des données, Nous allons effectuer les points suivants :

- Démontrer que l'accès à mariaDB est protégé par un login et un mot de passe.
- Démontrer que l'accès aux fonctionnalités de la base de données est sécurisé.
- Démontrer la sécurisation des données.
- Démontrer les autorisations et les limitations au niveau des actions de certains utilisateurs.

8.3 Les fonctionnalités

pour démontrer les fonctionnalités fournis par notre application de collecte. Nous allons effectuer les scénarios suivant :

8.3.1 Scénario (cf ANNEXE video_fonctionnalites)

- Un utilisateur quelconque, non administrateur essaie d’avoir accès à mariaDB.
- Un utilisateur quelconque, n’existant pas dans la base de données essaie de se connecter.
- Un utilisateur de la base de données utilise des credentials incorrectes pour se connecter
- Un utilisateur se connecte avec le rôle d’administrateur et crée un utilisateur avec le rôle médecin.
- Le médecin crée enregistre un patient.
- Le médecin crée enregistre une infirmière.
- Le médecin ou l’infirmière fait un examen sur un patient et collecte les informations.
- Le médecin ou l’infirmière modifie les données d’un examen.
- Le médecin ou l’infirmière affichage des données collectées .
- Le médecin ou l’infirmière affiche des données enregistrées pendant une période souhaitée.

Chapitre 9

Conclusion

Dans notre travail de Bachelor, nous avons présenté les différents types de bases de données, leurs avantages ainsi que leurs inconvénients. Nous avons par la suite, présenté les bases de données dans le domaine médicale, leur importance et les différents formats de données utilisés. Puis nous avons fait une analyse des besoins du client qui est la HESAV. Pour répondre à ces besoins, nous avons fait des choix techniques puis nous avons conçu une architecture adéquate modélisant le problème. Ensuite, nous avons présenté les différentes étapes de la conception de la base de données, ainsi que celle de l'implémentation de celle ci. Nous avons écrit des programmes Java pour l'importation des données depuis un fichier Excel et l'exportation de celles-ci dans un fichier Excel. Par la suite nous avons essayer de transformer les données de notre base en Common Data Model. En outre, Nous avons réalisé un API de collecte des données, fonctionnel et sécurisé. Cependant, Par soucis de temps, nous ne sommes pas aller jusqu'à la fin de la modélisation des fonctionnalités de l'interface graphique. En sommes, ce travail fut une très belle occasion d'apprendre, de mettre en pratique des connaissances acquises, de se familiarisé avec l'informatique du monde médical. Malgré les difficultés rencontrées, ce travail nous a par dessus tout, poussé au delà de nos limites par la recherche, l'exploration et l'apprentissage de nouvelles technologies.

Bibliographie

- [1] Diego Alvarez-Estevez. European data format, 2003.
- [2] auth0. Introduction to json web tokens, 2020.
- [3] EDUCBA. Difference between java ee and spring, 2020.
- [4] Google. Google trends compare angularjs vs reactjs vs vuejs, 2020.
- [5] Pedsnet. European data format, 2014.
- [6] Pedsnet. Framework, 2020.
- [7] Observational Health Data Sciences and Informatics. Ithe book of ohdsi, 2020.
- [8] SwissNeoNet. Aims and description, 2020.
- [9] Techopedia. Three-tier architecture, 2020.
- [10] Tutorialspoint. Apache poi - overview, 2020.
- [11] Wikipedia. Table(base de données), 2020.

Table des figures

2.1	Swissneonet	5
2.2	Pedsnet	6
4.1	MySQL vs MariaDB vs PostgreSQL vs MongoDB	12
4.2	MariaDB	13
4.3	Java EE vs Spring Boot vs PHP	14
4.4	Springboot	15
4.5	Angular vs React vs Vue.js	15
4.6	Google Trends : Angular vs React vs Vue.js dans le monde [4]	17
4.7	Google Trends : Angular vs React vs Vue.js dans le monde par pays [4] . . .	17
4.8	Google Trends : Angular vs React vs Vue.js en Suisse [4]	18
4.9	Google Trends : Angular vs React vs Vue.js en Suisse par canton [4]	18
4.10	Docker	19
5.1	Architecture 3-tier	20
5.2	Architecture du projet.	21
6.1	Database	23
6.2	Schéma UML	24
6.3	Table Institution	24
6.4	Table Role	25
6.5	Table User	25
6.6	Table Exam	25
6.7	Table Exam	26
6.8	Table DataCollectionType	26

6.9	Table Data	27
6.10	Table Modification	27
6.11	Relation entre les tables	28
6.12	Base de données modélisée	32
6.13	Résultat de l'importation des données	39
6.14	Données à exporter	39
6.15	Processus de transformation [7]	40
6.16	Tables dans de OMOP CDM	41
6.17	Mapping des colonnes	41
6.18	CDM transformation	42
6.19	Table Measurement	42
7.1	Interaction dans l'architecture	43
7.2	Structure du backend	44
7.3	Entities	45
7.4	Controllors	45
7.5	Services	46
7.6	Repository	46
7.7	Mapper	46
7.8	Interaction dans l'architecture	51
7.9	Login page	53
7.10	Register page	54
7.11	Reset-password page	54
7.12	Create an exam page	55
7.13	Data-type page	55
7.14	Five points of collect page	56
7.15	Child data form	56
7.16	Mother data form	57
7.17	Operation data form	57
7.18	Monitor data page	58

Liste des tableaux

2.1	Coût estimatif des besoins pour l'implémentation d'une base de données . .	5
2.2	Convention PCDM	6
3.1	Liste des données reçues	11
4.1	Tableau comparatif de quelques SGBD	13
4.2	Tableau comparatif de Java EE vs Spring/Spring Boot [3]	14
4.3	Tableau comparatif de Angular vs react vs Vue.js	16
B.1	Journal de travail	70

Annexe A

Diagramme de Gantt

N°	Titre	Travail demandé	Pré-requis	Janvier 2020 SEM 5, 27.01	Février 2020 SEM 6, 3.02 / SEM 7, 10.02 / SEM 8, 17.02 / SEM 9, 24.02	Mars 2020 SEM 10, 2.03 / SEM 11, 9.03 / SEM 12, 16.03 / SEM 13, 23.03	Avril 2020 SEM 14, 30.03 / SEM 15, 6.04 / SEM 16, 13.04 / SEM 17, 20.04 / SEM 18, 27.04	Mai 2020 SEM 19, 4.05 / SEM 20, 11.05 / SEM 21, 18.05 / SEM 22, 25.05	Juin 2020 SEM 23, 1.06 / SEM 24, 8.06 / SEM 25, 15.06 / SEM 26, 22.06 / SEM 27, 29.06	Juillet 2020 SEM 28, 6.07 / SEM 29, 13.07 / SEM 30, 20.07
0	Diagramme de Gantt Travail de Bachelor HEG + HESAV									
1	○	5,48 mois ? 24 févr. 2020			Diagramme de Gantt Travail de Bachelor HEG + HESAV...					
2	○	Reduction du rapport			Reduction du rapport					
3	○	24 févr. 2020			généralisé sur les bases de données					
3	○	3 jours ? 24 févr. 2020			type de base de données					
4	○	3 jours ? 26 févr. 2020			Base de données dans le medical					
5	○	1,4 semaines ? 28 févr. 2020			études des technologies					
6	○	6 jours ? 28 févr. 2020								
6	○	2,5 semaines ? 6 mars 2020								
7	○	Système UML								
8	○	1,5 semaines ? 23 mars 2020								
8	○	3 jours ? 1 avril 2020								
9	○	2,6 semaines ? 3 avril 2020								
10	○	2,6 semaines ? 13 avril 2020								
10	○	Test de la DB								
11	○	Implementation du Backend								
12	○	2,6 semaines ? 22 avril 2020								
13	○	Partie Authentification								
13	○	2,1sem? 6 mai 2020								
14	○	2 semaines ? 11 mai 2020								
14	○	Implementation des CRUD sur les données								
15	○	Test et validation								
15	○	1,1 mois ? 24 avril 2020								
16	○	3 semaines ? 26 mai 2020								
16	○	Implementation de l'interface graphique								
17	○	3 semaines ? 12 juin 2020								
17	○	Exportation des données sous GSV								
18	○	Exportation des données sous GSV								
18	○	2,6 semaines ? 19 juin 2020								
19	○	1,55 mois ? 5 juin 2020								
19	○	Test et finalisation								

Annexe B

Journal de travail

TABLE B.1: Journal de travail

Date	Description	Rech. [h]	Dev. [h]	Rapport [h]	Admin [h]
27.01.2020	Kick-off meeting avec le mandant HESAV	0	0	0	2
17.02.2020 - 11.03.2020	Recherche sur l'état de l'art	20	10	5	8
12.03.2020	Meeting avec la HESAV pour la mise au point et clarifications des incompréhensions. Présentation d'un schéma montrant le problème.	4	0	0	2
13.03.2020 au 25.04.2020	Modélisation en schéma UML, développement de la DB et documentation.	10	35	3	5
26.04.2020 au 14.05.2020	Test de laDB, mise en place de l'environnement de développement de l'API et documentation.	15	18	5	2
15.05.2020	Meeting avec la HESAV sur l'évolution du projet et clarification des incertitudes.	0	0	0	2
16.05.2020 au 10.06.2020	Gestion de la sécurité, importation et programmation des CRUD dans le backend.	20	80	22	10
10.06.2020 au 18.06.2020	Rédaction du rapport du travail intermédiaire.	10	15	30	4
19.06.2020	Finalisation du rapport et rendu du travail intermédiaire.	0	0	8	4
20.06.2020 au 17.07.2020	Développement de la partie interface, test et essai d'exportation des données.	40	110	15	10
20.07.2020 au 29.07.2020	Test, finalisation du rapport et relecture	5	20	60	10
30.07.2020	Test final, mis en page, et relecture du rapport	0	0	10	2
31.07.2020	Rendu du rapport, annexes, documents administratifs	0	0	4	1
	Total absolu de travail	124	288	159	62
	Pourcentage de travail	20%	45%	25%	10%

Annexe C

Concordance des noms

ID_Neo_Number_of_case_history	Case_history_number	Surname	Name	Gender	Age	Weight	Hight	Diagnosis	Starting_date_of_observation
ID (Neo_Number of case history)	Case history number	Surname	Name	Gender (0=male, 1=female)	Age (days of life)	Weight (g)	Hight (cm)	Diagnosis	Starting date of observation
Chirurgical_intevention_date	Anesthesia_type	HR_start	HR_induction	HR_traumatic	HR_after_op	HR_24h_after	BR_start	BR_induction	BR_traumatic
Chirurgical intevention date	Anesthesia type	HR (heart rate) beats per minute start	HR (heart rate) beats per minute induction	HR (heart rate) beats per minute traumatic	HR (heart rate) beats per minute after operation	HR (heart rate) beats per minute 24h after operation	BR (breasing rate) cycles per minute start	BR (breasing rate) cycles per minute induction	BR (breasing rate) cycles per minute traumatic
BR_after_operation	BR_24h_after_opera	Systolic_Blo	Systolic_Blo	Systolic_blo	Systolic_blo	Systolic_blo	Diastolic_blo	Diastolic_blo	Diastolic_blood_pressure_traumatic
BR (breasing rate) cycles per minute after operation	BR (breasing rate) cycles per minute 24h after opartion	Systolic blood pressure mmHg start	Systolic blood pressure mmHg induction	Systolic blood pressure mmHg traumatic	Systolic blood pressure mmHg after operation	Systolic blood pressure mmHg 24h after operation	Diastolic blood pressure mmHg start	Diastolic blood pressure mmHg induction	Diastolic blood pressure mmHg traumatic
Diastolic_blood_pressure_after	Diastolic_blood_pres	FIO2_start	FIO2_Inducti	FIO2_trauma	FIO2_After_o	FIO2_24h_A	SpO2_start	SpO2_induc	SpO2_traumatic
Diastolic blood pressure mmHg after operation	Diastolic blood pressure mmHg 24h after operation	FIO2 (O2 respiratory mixture)% start	FIO2 (O2 respiratory mixture)% induction	FIO2 (O2 respiratory mixture)% traumatic	FIO2 (O2 respiratory mixture)% after operation	FIO2 (O2 respiratory mixture)% 24h after operation	SpO2 (Saturation O2)% start	SpO2 (Saturation O2)% induction	SpO2 (Saturation O2)% traumatic
Spo2_after_operation	SpO2_24h_after_op	Hb_start	Hb_induction	Hb_traumatic	Hb_after_op	Hb_24h_afte	Ht_start	Ht_induction	Ht_traumatic
SpO2 (Saturation O2)% after operation	SpO2 (Saturation O2) % 24h after operation	Hb (hemoglobin)g/L start	Hb (hemoglobin)g/L induction	Hb (hemoglobin)g/L traumatic	Hb (hemoglobin)g/L after operation	Hb (hemoglobin)g/L 24h after operation	Ht (hematocrit) units*100, % start	Ht (hematocrit) units*100, % induction	Ht (hematocrit)units*100, %traumatic
Ht_after_operation	Ht_24h_after_opera	Er_start	Er_induction	Er_traumatic	Er_after_op	Er_24h_afte	Leik_start	Leik_inducti	Leik_traumatic
Ht (hematocrit)units*100, % after operation	Ht (hematocrit)units*10 0, % 24h after operation	Er (red blood cells)(units* 10^12)/L start	Er (red blood cells)(units* 10^12)/L induction	Er (red blood cells)(units* 10^12)/L traumatic	Er (red blood cells)(units* 10^12)/L after operation	Er (red blood cells)(units* 10^12)/L 24h after operation	Leik (white blood cells)(units* 10^9)/L start	Leik (white blood cells)(units* 10^9)/L induction	Leik (white blood cells)(units*10^9)/L traumatic
Leik_after_operation	Leik_24h_after_oper	Glucose_sta	Glucose_ind	Glucose_tra	Glucose_afte	Glucose_24h	K_start	K_induction	K_traumatic
Leik (white blood cells)(units*10^9)/L after operation	Leik (white blood cells)(units*10^9)/L 24h after operation	Glucose mmol/L start	Glucose mmol/L induction	Glucose mmol/L traumatic	Glucose mmol/L after operation	Glucose mmol/L 24h after operation	K (potassium) mmol/L start	K (potassium) mmol/L induction	K (potassium) mmol/L traumatic
Spo2_after_operation	SpO2_24h_after_op	Hb_start	Hb_induction	Hb_traumatic	Hb_after_op	Hb_24h_afte	Ht_start	Ht_induction	Ht_traumatic
SpO2 (Saturation O2)% after operation	SpO2 (Saturation O2) % 24h after operation	Hb (hemoglobin)g/L start	Hb (hemoglobin)g/L induction	Hb (hemoglobin)g/L traumatic	Hb (hemoglobin)g/L after operation	Hb (hemoglobin)g/L 24h after operation	Ht (hematocrit) units*100, % start	Ht (hematocrit) units*100, % induction	Ht (hematocrit)units*100, %traumatic