UNIVERSITE SAAD DAHLEB DE BLIDA

Faculté des sciences

Département d'informatique

MEMOIRE DE MASTER

En Informatique

Option : Ingénierie du Logiciel

THÈME :

Support de la variabilité dans les architectures orientées services

Réalisé par

EL MOHRI Meroua

BRIHI Fadhila

Encadré par

MADAME LAHIANI Nesrine

MADAME GUENDOUZ Amina

11 Juin 2023

Remerciements

C’est avec l’aide de Dieu qu’a vu le jour ce présent travail.

Ensuite, il n’aurait pas pu être achevé sans le soutien, les conseils et les encouragements de certaines personnes auxquelles nous tenons ici à exprimer nos sincères remerciements.

Tout d’abord, notre sincère reconnaissance et notre sincère gratitude à Madame LAHIANI Nesrine et Madame GUENDOUZ Amina, nos promotrices maîtres assistantes à l’Université SAAD DAHLAB de Blida, pour nous avoir proposé le sujet, pour leurs disponibilités, pour ses
précieux conseils, leurs confiances qu’elle nous a toujours témoigné et leurs sollicitudes dont elles nous a entouré, et ce tout au long de l’élaboration du présent travail.

Nous tenons à remercier chaleureusement nos enseignants de l’Université SAAD DAHLAB de Blida, pour le savoir qu’ils nous ont transmis toute au long de nos cursus universitaire. Un grand merci à tous ceux qui n’ont épargné le moindre effort, de près ou de loin, pour nous permettre d’accomplir notre projet.

Pour conclure, Nous adressons nos remerciements les plus respectueux au jury qui ont accepté d’évaluer notre travail.

Résumé

L’architecture orientée service est un style architecturale sur lequel on peut créer des services réutilisables, indépendants et interopérables. Dans cette architecture on peut fournir des services web
autonomes qui peuvent répondre au besoin et fonctionnalités d’un processus métiers pour fournir
un support efficace à la variabilité.
Le travail présenté propose une architecture orientée service, prenant en
charge l’automatisation de processus métier « traitement d’un dossier patient dans un hôpital ».
Cette solution mis en œuvre permettra d’informatisé toutes les fonctionnalités en rapport avec le dossier patient depuis l’admission à l’hôpital jusqu’à la sortie grâce aux
fonctionnalités offrir par Camunda sur les processus métiers et l‘architecture utilisé on peut rendre le système flexible et donner une meilleure vue sur le processus.

Mots clés :
L’architecture orientée service, service web, processus métier, Camunda, variabilité.

Abstract

Service-oriented architecture is an architectural style in which one can create reusable, independent and interoperable services, in this architecture one can provide autonomous services
which can take over the needs and functionalities of a business process, to provide effective support to variability

The work presented in this thesis proposes a service-oriented architecture, supporting the automation of business processes "the processing of a patient file in a hospital".

La solution mis en œuvre permettra d'avoir un dossier patient informatisable. Grâce Aux fonctionnalités offrir par Camunda sur les processus métiers et à l'architecture utilise en peut rendre le système flexible et donner une meilleure vue sur le processus.

Keywords :

Service-oriented architecture, web service, business process, Camunda, variability.

ملخص

الهندسة المعمارية الموجهة للخدمة هي أسلوب معماري يمكن من خلاله إنشاء خدمات قابلة لإعادة الاستخدام ومستقلة وقابلة للتشغيل البيني، في هذه البنية يمكن للمرء أن يوفر خدمات مستقلة يمكن أن تتولى احتياجات ووظائف عملية الاعمال، لتوفير دعم فعال للتنوع.
يقترح العمل المقدم في هذه الأطروحة بنية موجهة نحو الخدمة، تدعم اتمتة العمليات التجارية "معالجة ملف المريض في المستشفى"
الحل الذي يتم تنفيذه سيجعل من الممكن الحصول على ملف مريض محوسب.
وظائف التي تقدمها كاموندا في العمليات التجارية والبنية cجمال يمكن للوظائف المستخدمة ان تجعل النظام مرنا وتعطي رؤية أفضل للعملية

كلمات المفتاحية: البنية الموجهة تحت الخدمة، خدمة الويب، العمليات التجارية، كاموندا، التباين.

Liste des abréviations

BPD	**Business Process Diagram**
BPEM	**B**usiness **P**rocess **E**xecution **L**anguage
BPM	**B**usiness **P**rocess **M**anagement
BPMN	**B**usiness **P**rocess **M**odeling **N**otation
CORBA	**C**ommon **O**bject **R**equest **B**roker **A**rchitecture
DSML	**D**omain **S**pecific **M**odeling **L**anguage
FODA	**F**eature **O**riented **D**omain **A**nalysis
HTML	**H**yper **T**ext **M**arkup **L**anguage
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
LPL	**L**igne de **P**roduit **L**ogiciel
SOA	**S**ervice **O**riented **A**rchitecture
SOAML	**S**ervice **O**riented **A**rchitecture **M**odeling **L**anguage
SOAP	**S**imple **O**bject **A**ccess **P**rotocol
UML	**U**nified **M**odeling **L**anguage
WDSL	**W**eb **S**ervices **D**escription **L**anguage
XML	**E**xtensible **M**arkup **L**anguage

Table des matière

Table des figures

Liste des tableaux

Introduction Générale

Contexte de travail

La réutilisation des logiciels est un facteur clé pour les entreprises qui souhaitent augmenter la productivité des logiciels et améliorer la qualité, ainsi que réduire les coûts de développement et le temps de commercialisation. Ainsi, au lieu de tout développer à partir de zéro, il est préférable de réutiliser certains artefacts qui sont communs entre les applications. Actuellement, avec la croissance de la complexité et de la taille des logiciels, la maintenance et la compréhension des logiciels deviennent une tâche difficile. En outre, la flexibilité est un autre facteur important qui influe sur la réutilisation, car aujourd'hui, les packages d'applications pour grandes entreprises sont répandus, ce qui entraîne des problèmes chaque fois qu'un nouveau système est introduit ou que les besoins opérationnels nécessitent des changements dans les systèmes existants.

Dans le contexte de la réutilisation des logiciel, l'architecture orientée service (SOA) est un style Architectural qui favorise la création des systèmes flexibles et évolutifs en utilisant des services modulaire et interopérable, tandis que la variabilité est une réalité incontournable dans le développement des logiciels, car les besoins des utilisateurs et les exigences commerciales évoluent constamment. En outre SOA offre un support puissant pour la gestion de la variabilité, permettant aux organisations de concevoir et de développer des systèmes logiciels adaptable et réutilisable.

Problématique

Une étude approfondie sur les techniques de développement de systèmes autonomes et auto adaptatifs et leur application dans l'amélioration des capacités dynamiques de produit nousa permis de retirer quelques conclusions relatives aux limites de ce domaine, celles-ci peuvent se résumer dans les points suivants :

- Les approches actuelles ont présenté uniquement la relation entre les concepts de SPL et de SOA plus précisément au niveau conceptuel, mais sans entrer dans les détails de la mise en œuvre.
- La variabilité, même si elle existe dans les services ou les processus métier au sein d'une architecture orientée services, est implicite ce qui nous empêche de la gérer convenablement dans le but d'améliorer la réutilisation.

- Les approches actuelles se concentrent sur les services en tant qu'artefact de base dans SPL dans lequel les services gèrent des variabilités,

Ceci nous amène à nous poser certaines questions cruciales qui :

QR1 : Comment gérer et modéliser la variabilité dans l'architecture orientée service ?

QR.2 : Quelles sont les technologies telles que les modèles de services, les Frameworks, les langages et les outils utilisés pour la conception et le développement de SOA ? Comment faciliter la sélection des technologies qui répondent à un objectif particulier ?

Objectifs du travail

L'objectif de notre travail consiste à :

• Progresser l'état de l'art en matière de conception et de mise en œuvre de la variabilité

• Proposer des techniques de modélisation nécessaires pour représenter explicitement la variabilité dans les architectures orientées services.

• Concevoir et développer un système médical qui se repose sur un processus automatisés flexibles.

Organisation du mémoire

Ce mémoire est composé d'une introduction générale suivie de deux parties, une partie

Théorique et une autre pratique.

- **Chapitre 1 :** La première partie sera consacrée à l'étude théorique des concepts principaux rencontrés lors de la réalisation de notre projet, structurée en trois sections. La première section va traiter la modélisation de la variabilité, la seconde s'intéressera à l'architecture orientée service et à la notion de processus métier. Enfin, la troisième section présentera les travaux connexes.

- **Chapitre 2 :** Après avoir donné une idée sur les concepts utiles pour la réalisation de notre projet, nous aborderons dans la deuxième partie, le développement de notre solution, adoptée pour la mise en œuvre d'une architecture orientée service, en utilisant les langages de modélisa- tion UML et BPMN. Nous verrons dans un premier temps, La conception suivi de la modélisation de notre solution dans laquelle on a amélioré la notion de BPMN et les processus métiers en ajoutons les stéréotypes pour représenter la notion de la variabilité.

- **Chapitre 3 :** En dernier, nous passons à l'implémentation de la solution retenue au niveau de la conception après une description de l'environnement du travail. Nous achèverons cette partie avec la présentation de système réalisée. Puis une conclusion générale qui vienne pour clôturer ce mémoire.

Chapitre 1

Etat d'art

1.1 Introduction :

Ce chapitre est consacré à l'étude approfondie de notre projet. Afin de fournir une analyse Complète, nous avons structuré ce chapitre en trois sections distinctes. La premier section nous a permis de bien comprendre la variabilité et sa modélisation, la deuxième section nous a expliqué l'architecture orientée service et les processus métiers, et la troisième section a mis en lumières les travaux existants et les opportunités de recherche future.

1.2 La modélisation de la variabilité

1.1 La variabilité

La variabilité fait référence à la capacité de modifier ou personnaliser un système. Ce concept est lié à celui de «point de variation ». Un point de variation identifie et localise l'endroit où se produit la variabilité. Il précise les solutions possibles de résolution de cette variabilité (explicitation des variantes), et éventuellement le moment de cette prise de décision (en anglais "Binding time"). [5]

1.2 Catégorie de la variabilité :

Des approches tendent à réaliser des distinctions entre des catégories de variabilité. Concernant le Niveau des exigences, Halmans et Pohl [14] font une distinction entre la variabilité essentielle ("Essential") et technique ("technical"). La variabilité essentielle est celle dédiée au point de vue Client, définissant et délimitant les artefacts à implémenter. La variabilité technique correspond Quant à elle à une utilisation par les ingénieurs, définissant de ce fait, la manière d'implémenter les artefacts. D'autre part, et dans la continuité des travaux de Pohl et al [18] et Metzger et al [2] proposent de scinder la variabilité en deux : (i) la variabilité de la ligne de produit ("Product line variabilité"), et (ii) la variabilité du logiciel ("software variability"). La première réfère aux variations entre les systèmes appartenant à une LDP en termes de propriétés et qualités, la seconde relève de l'habileté d'un système logiciel à être efficacement étendu, modifié, spécialisé ou configuré pour un besoin donné.

1.3 Granularité de la variabilité :

Kircher et al rapportent sur l'importance de séparer l'espace du problème de celui de la solution [21]. Traitant des exigences, ces derniers insistent sur la séparation à réaliser entre des exigences clients et des exigences techniques de plus bas niveaux. De leur côté, Kim et al [27] évoquent une variabilité conventionnelle et une variabilité des composants. Les auteurs présentent cinq types de variabilité : (i) des attributs, (ii) de logique (algorithmique), (ii) de flux ("workflow"),

(iv) de persistante, et (v) d'interface. Etxeberria et al [9] Explicitent et distinguent les features fonctionnels de ceux de relatifs à La qualité. Néanmoins, Moreira et al défendent, dans le domaine des exigences, que la séparation des préoccupations dans les modèles de features ne peut être restreinte à une bi-dimensionnalité fonctionnelle / non-fonctionnelle, les deux dimensions étant transverses [1]. Les auteurs avancent une division en méta concerns (préoccupations génériques et communes à plusieurs Systèmes) et system concerns (préoccupations spécifiques à un système donné).

1.4 Différents modelés de variabilités :

Cette section offre une vue d'ensemble des modèles de variabilité ont présente la variabilité indépendante et la variabilité couplée aux artefacts :

- Une représentation de la variabilité indépendante :

Cette catégorie regroupe les méthodes dans lesquelles l'expression de la variabilité reste indépendante aux artefacts. Ces approches sont pour la plupart anciennes, les approches plus récentes visent un automatisme et tentent d'assurer et d'optimiser les liens vers des artefacts de la solution.

-Les méthodes basées documentation :

En peut présenter la variabilité avec des outils comme les feuille Excel.

-Les méthodes basées "feature diagram" :

Ces méthodes mettent en œuvre des diagrammes de caractéristiques indépendants des

Artefacts de conception, à savoir le code, la documentation et les modèles. Ce type d'approche naît avec l'apparition de la méthode d'analyse FODA [16] et tend à disparaitre au profit de méthodesd'avantage orientées vers l'implémentation.

- Une représentation de la variabilité couplée aux artefacts : Tel que relevé par Haugen et al [31], Il y a deux catégories de techniques qui introduisent la variabilité dans les langages de modélisation : une approche de fusion ("amalgamated") et de séparation ("separated").

-La variabilité fusionnée : Ce type d'approche propose d'augmenter les capacités d'un langage par des concepts de variabilité. Ces langages peuvent être des DSMLs ou un langage générique de méta-modélisation, typiquement UML.

-La variabilité séparée : Dans cette catégorie d'approches, une séparation de la

Préoccupation de modélisation de la variabilité est clairement établie, en effet, les méta-modèles des

Artefacts de conception et du langage de variabilité sont indépendants. [5] comme montre la figure suivant :

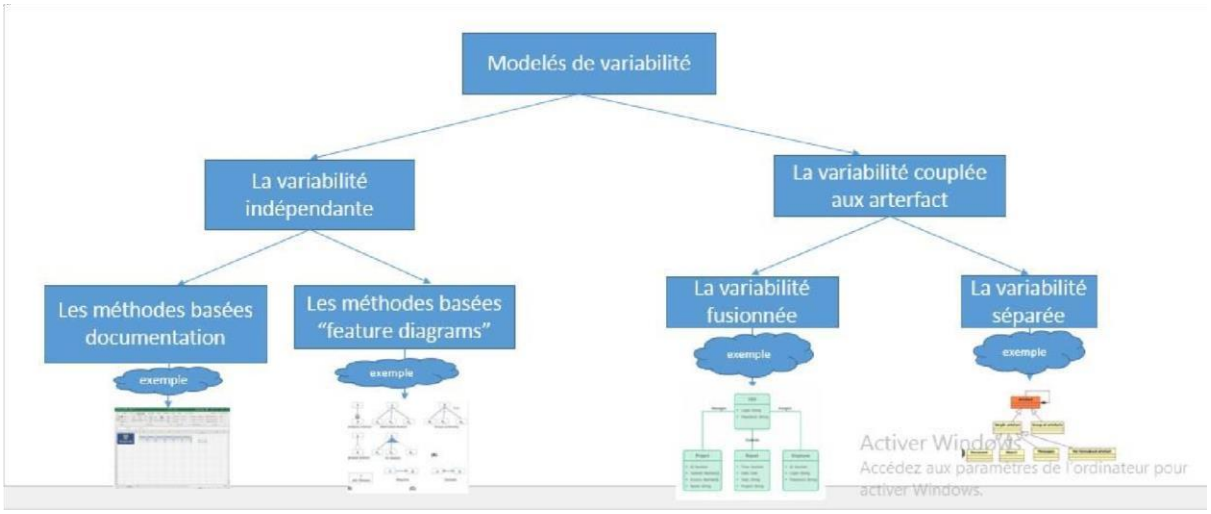


FIGURE 1.1 – Modèle de variabilité

1.5 Les lignes de produits logiciels : Suite aux premiers travaux sur la réutilisation proposés par McIlroy dans [20], David Parnas introduit le concept de « Program Families » dans [25], où il définit une ligne de produits (LDP) ou « Program Family», comme « un ensemble de programmes dont les propriétés communes sont si nombreuses que c'est avantageux d'étudier d'abord ses propriétés communes avant d'analyser les membres individuels de l'ensemble » C.-à-d. on peut définir ces lignes de produit comme une famille de produits logiciels dont la production est encadrée à travers une architecture générique et un ensemble d'artefacts logiciels communs et réutilisables.

• **Processus et méthodologie :** L'ingénierie des LPLs est un paradigme de développe- ment qui se décompose en deux processus sont l'ingénierie du domaine, suivie de l'ingénierie d'application. [3]

1/l'ingénierie du domaine : durant laquelle les éléments communs et les variations au sein de la famille de produits sont caractérisés et implémentés. L'objectif est donc le développement pour la réutilisation. Ce processus se décompose en trois phases, illustrées en Figure 1.1 [3] :

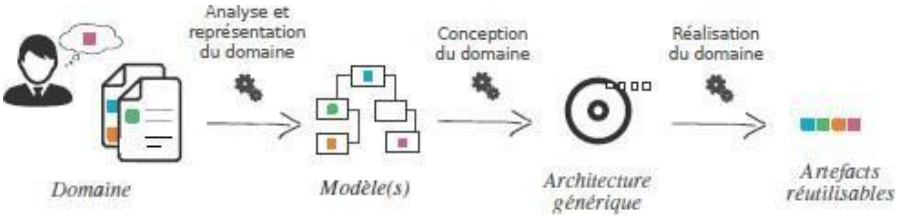


FIGURE 1.2 – l'ingénierie du domaine : le développement pour la réutilisation [3]

2/l'ingénierie d'application : correspond cette fois au développement par la réutilisation. Son objectif est d'exploiter la variabilité mise en place lors de l'ingénierie du domaine pour Faciliter le développement des variantes logicielles de la LPL. On distingue deux phases, qui sont illustrées en Figure 1.2 [3]

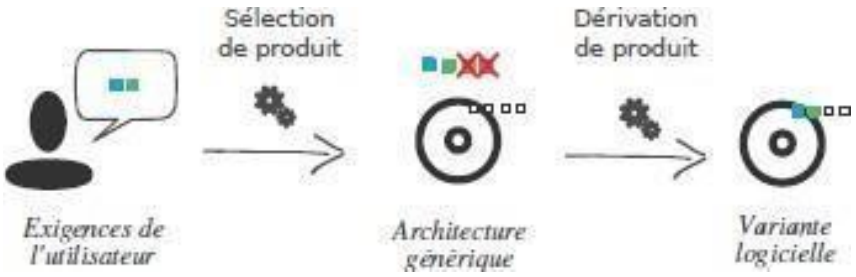


FIGURE 1.3 – l'ingénierie d'application : le développement par la réutilisation [3]

1.3 Architecteur orientée service et Processus métiers

1.2.1 –L'architecture orientée service

En considérant le terme architecture orientée services, il est utile de définir les termes clés Suivant :

- Une architecture est une description formelle d'un système, définissant son objectif, ses fonctions, ses propriétés visibles de l'extérieur et ses interfaces.

- Un service est un composant logiciel auquel on peut accéder via un réseau pour fournir une fonctionnalité à un demandeur de service.

- Le terme architecture orientée services fait référence à un style de construction de systèmes distribués fiables qui fournissent des fonctionnalités en tant que services, avec l'accent supplémentaire sur le couplage lâche entre les services en interaction.

1.2.2 Caractéristique de service :

Nous considérons SOA comme un style architectural qui met l'accent sur la mise en œuvre de composants en tant que services modulaires pouvant être découverts et utilisés par les clients.

Les services ont généralement les caractéristiques suivantes :

- Les services peuvent être utiles individuellement, ou ils peuvent être intégrés—composés—pour fournir des services de niveau supérieur. Cela favorise la réutilisation des fonctionnalités existantes.

- Les services communiquent avec leurs clients en échangeant des messages : ils sont définis par les messages qu'ils peuvent accepter et les réponses qu'ils peuvent apporter. - Les services peuvent participer à un workflow.

- Les services peuvent être complètement autonomes ou dépendre de la disponibilité d'autres services ou de l'existence d'une ressource telle qu'une base de données.

- Les services annoncent des détails tels que leurs capacités, leurs interfaces, leurs politiques et les protocoles de communication pris en charge. Les détails de mise en œuvre ne concernent pas les clients et ne sont pas révélés.

1.2.3 Les technologies de SOA :

Bien que beaucoup ait été écrit sur SOA et les services web, il existe encore une certaine confusion entre ces deux termes parmi les développeurs de logiciels. SOA est un style architectural, tandis que les services Web sont une technologie qui peut être utilisée pour mettre en œuvredes SOA. La technologie des services Web se compose de plusieurs normes publiées, les plus importantes étant SOAP et WSDL. D'autres technologies peuvent également être considérées comme des technologies de mise en œuvre SOA, telles que CORBA. Bien qu'aucune technologieactuelle ne réponde entièrement à la vision et aux objectifs de SOA tels que définis par la plupartdes auteurs, elles sont toujours appelées technologies SOA. La relation entre les technologies SOA et SOA est représentée dans la figure 1.3

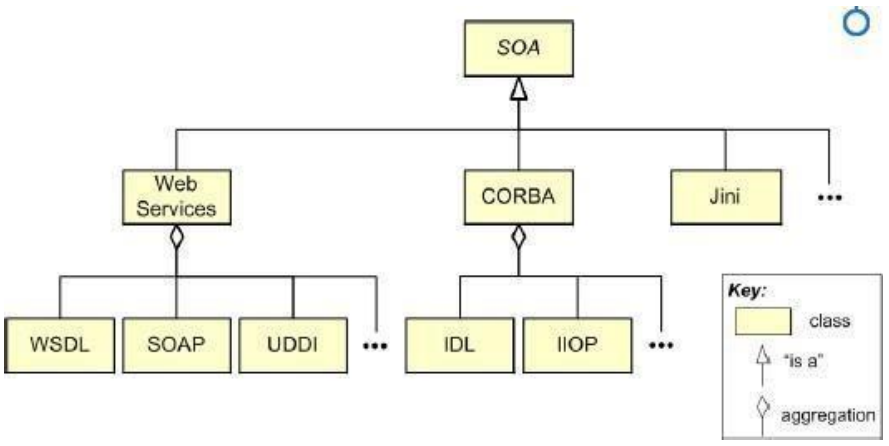


FIGURE 1.4 – Les technologies de SOA [4]

1.2.3.1 Services Web : Les services Web sont des composants logiciels distribués qui fournissent des informations aux applications plutôt qu'aux humains, via une interface orientée application. Les informations sont structurées à l'aide du langage XML (Extensible Markup Language), de sorte qu'elles peuvent être analysées et traitées facilement plutôt que d'être formatées pour l'affichage.

Les principales spécifications utilisées par les services Web sont :

- XML (Extensible Markup Language) un langage de balisage pour le formatage et l'échange de données structurées.

- SOAP (à l'origine Simple Object Access Protocol, mais techniquement plus un acronyme) un protocole basé sur XML pour spécifier les informations d'enveloppe, le contenu et les informations de traitement d'un message.

- WSDL (Web Services Description Language) un langage basé sur XML utilisé pour décrire les attributs, interfaces et autres propriétés d'un service Web. Un document WSDL peut

être lu par un client potentiel pour en savoir plus sur le service. Bien qu'un service Web puisse prendre en charge n'importe quel protocole de communication et offrir un choix à ses clients, le plus courant est SOAP sur HTTP ou HTTPS. Cela contribue à l'attrait des services Web, car HTTP et HTTPS sont présents et ne posent généralement pas de problèmes de traversée de pare-feu dans une organisation qui autorise le trafic HTTP bidirectionnel.

1.2.2. Les processus métiers :

1.2.2.1. Définition de processus :

Un processus est défini comme "une séquence d'événements naturels qui se produit dans le même ordre », il est définit dans le domaine technique comme « une série d'opérations conduisant à des résultats », en particulier dans le processus de fabrication d'expression [30]

1.2.2.2. Définition de processus métiers :

Le processus métier ou "business process" en anglais se définit par la gestion du workflow Management Coalition (WfMC) comme :"un ensemble de procédures ou d'activités liées les unes aux autres pour atteindre collectivement un objectif métier en définissant les rôles et les interactions fonctionnelles au sein d'une structure organisationnelle". En termes simples, il est défini comme une séquence d'activités menant à un résultat commercial. Comme illustre la figure 1.4

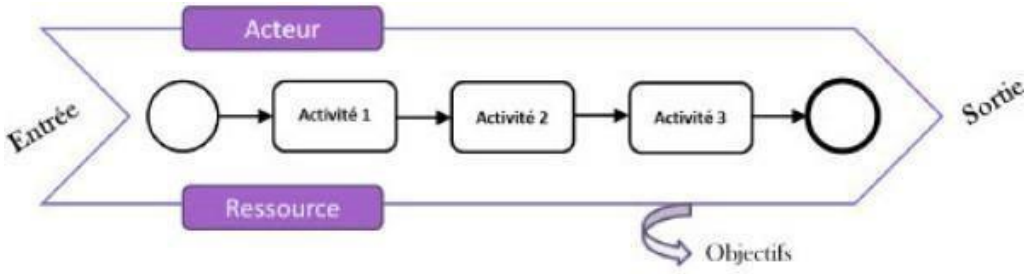


FIGURE 1.5 – le fonctionnement d'un processus métier [24]

1.2.2.3. Gestion des processus métier (BPM)

1.2.2.3.1. Définition de la Gestion des processus métier (BPM) : Il existe plusieurs définitions de BPM (business process management en anglais), y compris la définition de Van der Aalst: «Soutenir les processus métier à l'aide des méthodes, des techniques et des logiciels pour concevoir, mettre en œuvre, contrôler et analyser des processus opérationnels impliquant des humains,des organisations, des applications, des documents et d'autres sources d'informations»[10]. Sonbut est de réaliser des gains significatifs d'efficacité et de productivité. Les objectifs du BPM sont [11] :

- Automatiser les processus d'affaires de l'entreprise
- Éliminer autant que possible les temps d'attente dans ces processus

- Accélère considérablement la phase de prise de décision en fournissant des informations en temps réel et en s'adressant à la bonne personne

1.2.2.3.2. La notation BPMN 2.0

Afin de documenter les processus métiers, une manière appropriée est d'utiliser une notation, souvent graphique, pour les modéliser. Une notation graphique adaptée est un moyen de casser cette frontière par un langage compréhensible par tous les maillons de la chaîne, ce qu'on appelle BPMN acronyme pour « Business Process Modeling Notation » BPMN est une norme pour La modélisation des processus métier qui fournit une notation graphique pour spécifier les processus métier dans un Diagramme de Processus métier (BPD), basé sur les techniques traditionnelles d'organigramme [26]. L'objectif principal de BPMN est de fournir une notation qui est compréhensible par chaque intervenant du projet informatique : de l'analyste métier qui rédige la version initiale du processus, au client qui validera le processus, au développeur technique qui est responsable de l'implémentation technologique du logiciel qui exécutera ces processus puis finalement aux manageurs qui géreront et contrôleront l'efficacité du processus [29].

BPMN 2.0 contient ces cinq types d'éléments [12] :

1. Les objets de flux (Flow Object) ;
2. Les données (data) ;
3. Les objets de connexion (Connecting Object) ;
4. Les partitions (Swimlanes) ;
5. Les artéfacts (Artifacts) ;

1. Les objets de flux : Les objets de flux sont les principaux éléments graphiques permettant de définir le comportement d'un processus métier. Il existe trois objets de flux « les activités », « Les événements » et « les passerelle » :

1-Activities : une activité est un travail effectuée dans le cadre d'un processus métier. Il est représenté par un rectangle aux coins arrondis. Qui correspondent à une action qui peut être réalisée par un humain ou une machine. Il peut être atomique ou non atomique (composé). Ils peuvent être classés en trois types : tâche, sous-processus et activité d'appel.

-Tâche : une tâche est une activité atomique incluse dans un processus. Une tâche est utilisée lorsque le travail dans le processus ne peut pas être décomposé à un niveau de détail plus fin. Un objet tâche partage la même forme que le sous-processus, qui est un rectangle aux coins arrondis.il existe sept types de tâches et dans le tableau suivant on a expliqué quelque tâches :

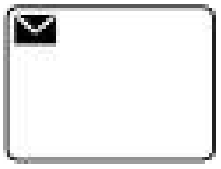
Elément	Description	Notation
Tâche service	est une tâche qui peut être un service Web ou une application automatisée.	
Tâche d'envoi de message	est une tâche pour envoyer un message à un participant externe au processus.	
Tâche de réception d'un message	est une tâche pour attendre l'arrivée d'un message d'un participant externe au processus. Une fois le message reçu, la tâche est terminée.	
Tâche utilisateur	est une tâche utilisée pour modéliser le travail qui doit être effectué par un acteur humain	
Tâche manuelle	est une tâche utilisée pour modéliser le travail effectué par quelqu'un que le moteur n'a pas besoin de connaître et qui n'a pas de système ou d'interface utilisateur connu	

TABLE 1.1 – Tableau des tâche BPMN.

-Sous-processus : est une activité composée dont les détails internes ont été modélisés à l'aide d'activités, de passerelles, d'événements et de flux de séquence. C'est un objet graphique dans un processus, mais il peut également être "ouvert" pour afficher un processus de niveau inférieur, un sous processus est représentée par une tâche avec un petit (+) permettant d'accéder au détail (sous-processus réduit), ou représenté dans sa version élargie, comme le montre la figure 1.6

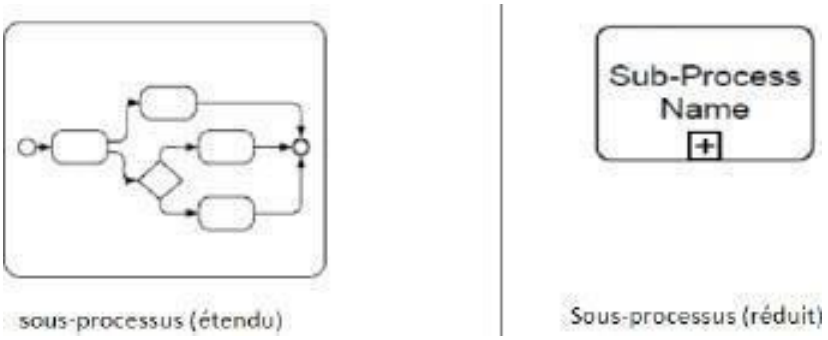


FIGURE 1.6 – Les formes de sous-processus [24]

Il existe cinq types de sous-processus : Loop, Multi-instance, Compensation, Ad hoc et Compensation et Ad-Hocc, comme montre l'image.

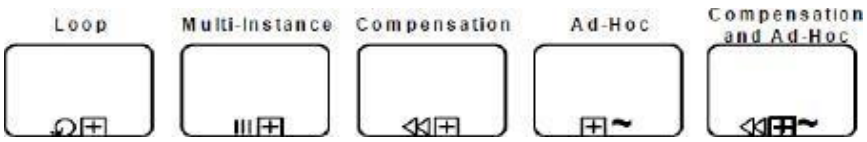


FIGURE 1.7 – Les types de sous-processus. [24]

2 - Evénements : un événement est quelque chose qui "se produit" au cours d'un processus.
Il existe trois types d'événements, en fonction du moment où ils affectent le flux : début, intermédiaire et fin. Comme montre la figure suivant :

FIGURE 1.8 – Les évènement dans BPMN 2.0. [24]

Il existe d'autre types d'évènements de BPMN2.0 sont représentée par la figure suivante

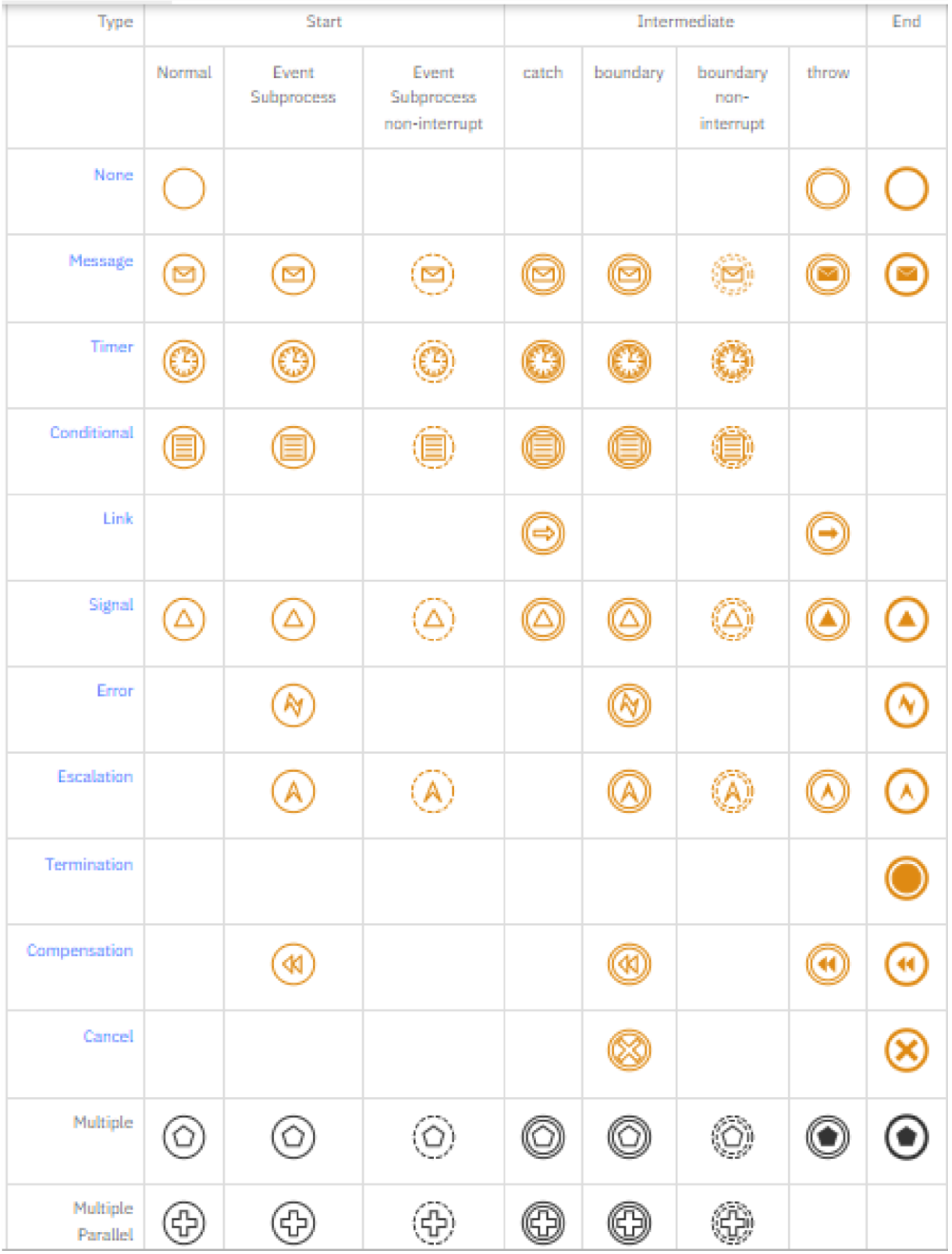


FIGURE 1.9 – Les types d'événements de BPMN2.0. [6]

3- Les passerelles(Gateway) : les passerelles sont utilisées pour contrôler la façon dont les flux de séquence interagissent lorsqu'ils convergent et divergent au sein d'un processus. Le terme

« Passerelle » implique qu'il existe un mécanisme de déclenchement qui autorise ou interdit le passage par la passerelle. Les différentes passerelles possibles sont :

FIGURE 1.10 – Les passerelles de BPMN2.0. [24]

2/Les données : Les données sont des éléments physiques ou des informations qui sont créés, manipulés ou utilisés lors de l'exécution d'un processus.

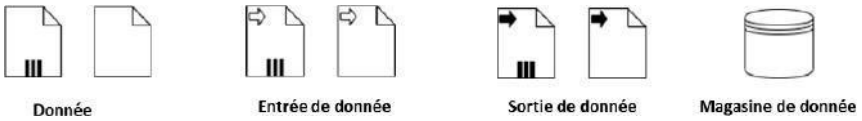


FIGURE 1.11 – Les données de BPMN 2.0 [24]

3/Les objets de connexion

• **Flux de séquence :** un flux de séquence est utilisé pour connecter des objets de flux et pour afficher l'ordre des éléments de flux dans un processus. La source et la cible de flux de séquence doivent appartenir à l'ensemble des éléments de flux suivants : événements, activités, et passerelles BPMN 2.0 utilise cinq (4) types de flux de séquence.

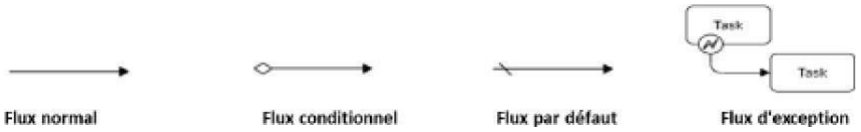


FIGURE 1.12 – Les flux de séquence de BPMN 2.0 [24]

• **Les flux de message (Message Flow) :** Les flux de messages assurent l'envoi de messages d'une entité à une autre. Ils doivent connecter un pool ou un objet se situant dans ce pool à un autre pool ou un objet se situant dans ce dernier.

FIGURE 1.13 – Les flux de message de BPMN 2.0. [24]

4. Les partitions (SWIMLANE) :

• **Les participant (Pool) :** est la représentation graphique d'un participant dans une collaboration. Il agit également comme un « SWIMLANE » et un conteneur graphique pour partitionner un ensemble d'activités d'autres pools.

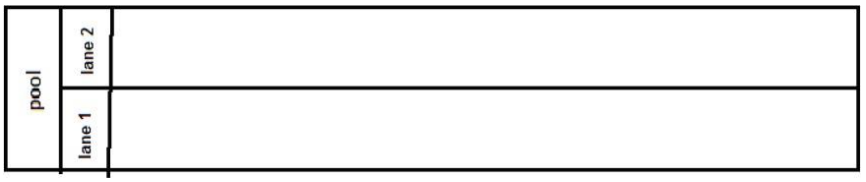


FIGURE 1.14 – La représentation d'un pool en BPMN 2.0. [24]

• **Les couloirs (Lane) :** une voie est une sous-partition au sein d'un processus, parfois au sein d'un pool, et s'étendra sur toute la longueur du processus, soit verticalement, soit horizontalement. Les couloirs sont utilisés pour organiser et catégoriser les activités.

FIGURE 1.15 – La représentation d'une LANE en BPMN 2.0 [24]

1.4 Travaux connexe :

Dans ce qui va suivre, nous allons analyser quelques travaux de la littérature qui traitent la modélisation de la variabilité dans les architectures orientées service.

Travail de : « Tuan Nauyen et al» [28]

Ils ont défini un middleware génératif qui prend en charge le déploiement de service et exploite le mappage pour permettre la personnalisation du service d'exécution. Ils ont utilisé Amazon Web Services pour l'évaluation. En plus de relever le défi de fournir des services personnalisables, Leur expériences montre que le middleware génératif aide à réduire la consommation de ressources d'exécution.

Travail de : « Mohammad Abu-Matar et al» [23]

Dans cette étude, ils ont décrit des scénarios de variabilité SOA en utilisant le nouveau standard OMG SoaML, ils ont développé un méta-modèle qui cartographie les caractéristiques SoaML. Ils ont supposé qu'une telle approche facilite la gestion de la variabilité des familles de services dans une manière systématique et indépendante de la plate-forme.

Travail de : « Juanjuan Jiang et al» [15]

Juanjuan Jiang et all ont proposé une catégorisation des points de variation dans les points de terminaison de service, les descriptions WSDL et la logique métier. Leur approche est basée sur des modèles pour gérer la variation et spécifier un cadre de service.

Travail de : «Maryam Razavian et Ramtin Khosraviss » [19]

Razavian et all ont proposé une méthode de modélisation de la variabilité dans les modèles de processus métier. Ils ont conçus une méthode basé sur UML2, ils ont également étudient des moyens pour éviter d'encombré le modèle et réduire leur complexité.

Travail de : « M.Koning et al » [22]

Dans cette étude, ils ont présenté VxBPEL qui est une extension de langage BPEL , ils ont développé un Prototype pour prendre charge de déploiement, l'exécution et la reconfiguration de processus variable, ils ont développé un service web dynamique.

Approche	Variability Modelling	Compatibility	Extensibility	Consistency	Application domain	Tool support
[28]	Feature Modèle	Oui	Oui	Oui	Swinsure Amazon -Web Service	Service capability modeling tool
[23]	SoaMl	Oui	Oui	Oui	E-commerce	Aucun
[15]	SOAP, WSD L	Oui	Oui	Oui	référentiel simple services	Aucun
[19]	UML2	Oui	Oui	Oui	Système d'approvisionnement	Aucun
[22]	VXBPEL	Oui	Oui	Oui	Système d'approbation	ActiveBPEL

TABLE 1.2 – Tableau comparative.

1.5 Discussion :

Les travaux présentés dans le tableau précédant proposent des solutions pour la

prise en charge de la variabilité dans mes SOAs, Cependant il y’a des lacunes que

Nous allons citer dans cette section. Dans l'article de Tuan Nauyen et al qui ont défini un middleware génératif prend en charge le déploiement de service et exploite le mappage pour permettre la personnalisation du service d'exécution, Nous remarquons l'absence de supports de variabilité dans le modèle de processus métier .Dans l'article de Mohammad Abu-Matar et al qui ont décrit des scénarios de variabilité SOA en utilisant le nouveau standard OMG SoaML , nous observons le manque de la gestion des règles de transformation de la variabilité . Juanjuan Jiang et al ont proposé une catégorisation des points de variation, mais sans prendre en charge l'aspect statique de la gestion de la variabilité. Maryam Razavian et Ramtin Khosraviss ont proposé une méthode de modélisation de la variabilité dans les modèles de processus métier, mais leur approche ne supporte pas tous les types de variabilité (OR et VP Ouvert).En fin M.Koning et al ont présenté VxBPEL qui est une extension de langage BPEL. Dans leurs proposition il manque certain types variabilité (Optionnel et ouvert), en plus de l'absence de la possibilité de modifier les configurations des processus en cours d'exécution et le manque des contraintes (inclusion et extensions). Pour conclure, malgré les efforts fournis dans ce domaine, les travaux existant présents toujours des limites qui doivent être surmonté afin de bénéficier plus d'avantages des SOA et de la gestion de la variabilité.

1.6 Conclusion

Ce chapitre nous a fournir une base solide pour aborder le chapitre suivant sur l'analyse et la conception de la solution .les trois sections nous ont apporté une compréhension approfondie sur la modélisation de la variabilité, l'architecture orientée service, les processus métier et les travaux connexe. Maintenant, nous somme prêt à appliquer ces connaissances pour analyse et concevoir des solutions qui repondent aux besoins spécifiques à notre projet.

Chapitre 2

Analyse et Conception de la solution

2.1 Introduction :

Nous présentons dans ce chapitre la partie analyse et conception de notre solution qui consiste à informatisé le dossier patient en particulier au sein de l'Hôpital DR FARES YAHIA de la wilaya de TIPAZA. Nous avons choisis de suivre un processus de développement qui répond à nos objectifs. Dans ce chapitre nous commençons par motiver notre travail. Ensuite, nous présentons le processus de développement, puis nous entamons l'analyse de besoin et la modélisation toute en montrant notre contribution.

2.2 Motivation et cas d'étude :

Malgré les efforts déployé par nos institues gouvernementales notamment les hôpitaux pour informatiser les processus métier et éviter la bureaucratie, nous trouvons toujours les patients confrontés à un processus lourd de paperasses et de déplacement d'un service à un autre pour prendre en charge leurs dossiers. Tenons en compte l'état dans lequel soit un patient (qui est malade ou même les membres de sa famille) dans un hôpital, cela nous amène à penser à une manière qui exempte le patient de cette tâche toute en assurant la rapidité et l'efficacité de la gestion des dossiers des patients.

Le but de notre travail est principalement d'informatiser le processus de traitement d'un dossier de patient depuis son entrée à l'hôpital jusqu'à sa sortie, en prenant en charge les différent types de patient ainsi que les différent services de l'hôpital et en minimisant le plus possible les activités manuelles. Et cela va être bénéfique à la fois pour les patients et les employeurs de l'hôpital.

2.3 Méthodologie de développement :

Dans notre projet, nous avons contribué à un processus appelé le processus de développement.
Ce processus commence par une discussion approfondie avec des experts du domaine (ingénieur et médecin) qui dois aider à générer des diagrammes d'analyse de besoin (diagramme de cas d'utilisation) et des décisions pour la conception et l'implémentation.
L'étape suivante consiste à concevoir le système en utilisant deux langages de modélisation principaux UML pour le diagramme de classe et BPMN pour modéliser le processus métier. La modélisation doit baser sur le concept de service web et prendre en charge la modélisation de la variabilité.
Dans les étapes suivantes, le système doit être implémenté puis tester et valider par un / des expert du domaine.

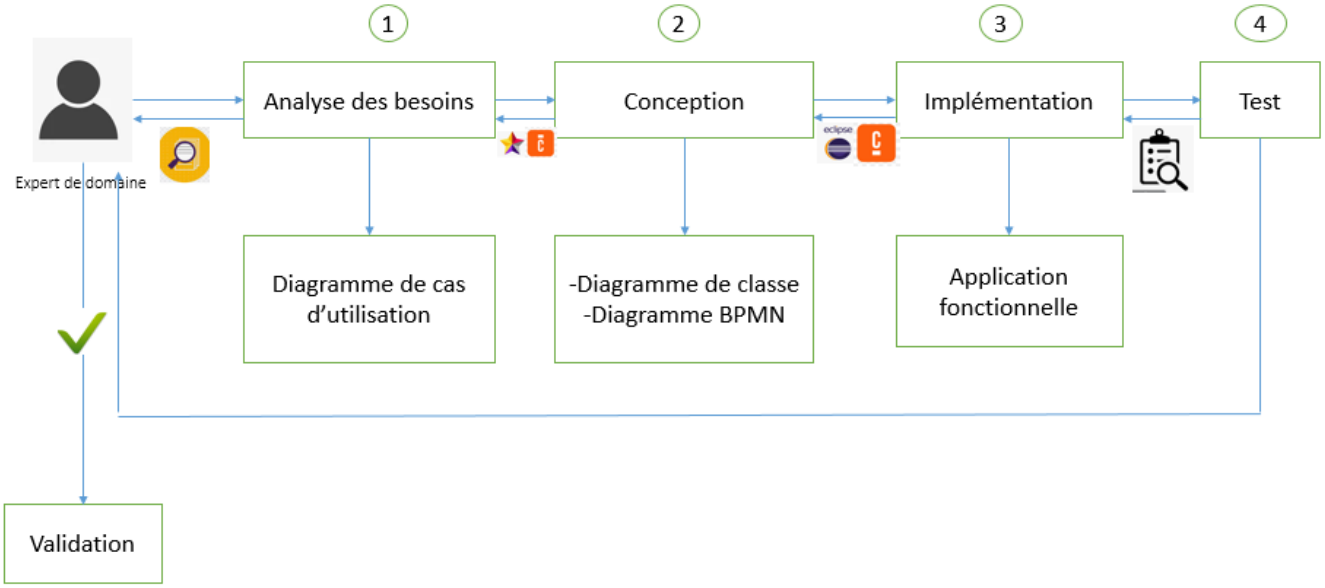


FIGURE 2.1 – Processus de développement

2.4 Analyse des besoins :

2.4.1 Diagramme de cas d'utilisations :

Avant la modélisation des processus métiers, on doit identifier les acteurs et spécifier les exigences minimales.

A. Identifications des acteurs :

Nous avons comme acteurs : médecin, secrétaire, administrateur et infirmier.

B. Spécification des besoins :

Table 2.1 : Attribution des rôles et des fonctionnalités

auteur	besoins
médecin	Gérer une consultation Gérer une hospitalisation Gérer traitement Saisir un bon de traitement médical
secrétaire	Gérer une fiche navette Gérer un rapport
administrateur	Gérer personnelle Gérer service Gérer matériel Gérer spécialités
infirmier	Gérer constants Gérer soins

C. Diagramme des cas d'utilisation général :

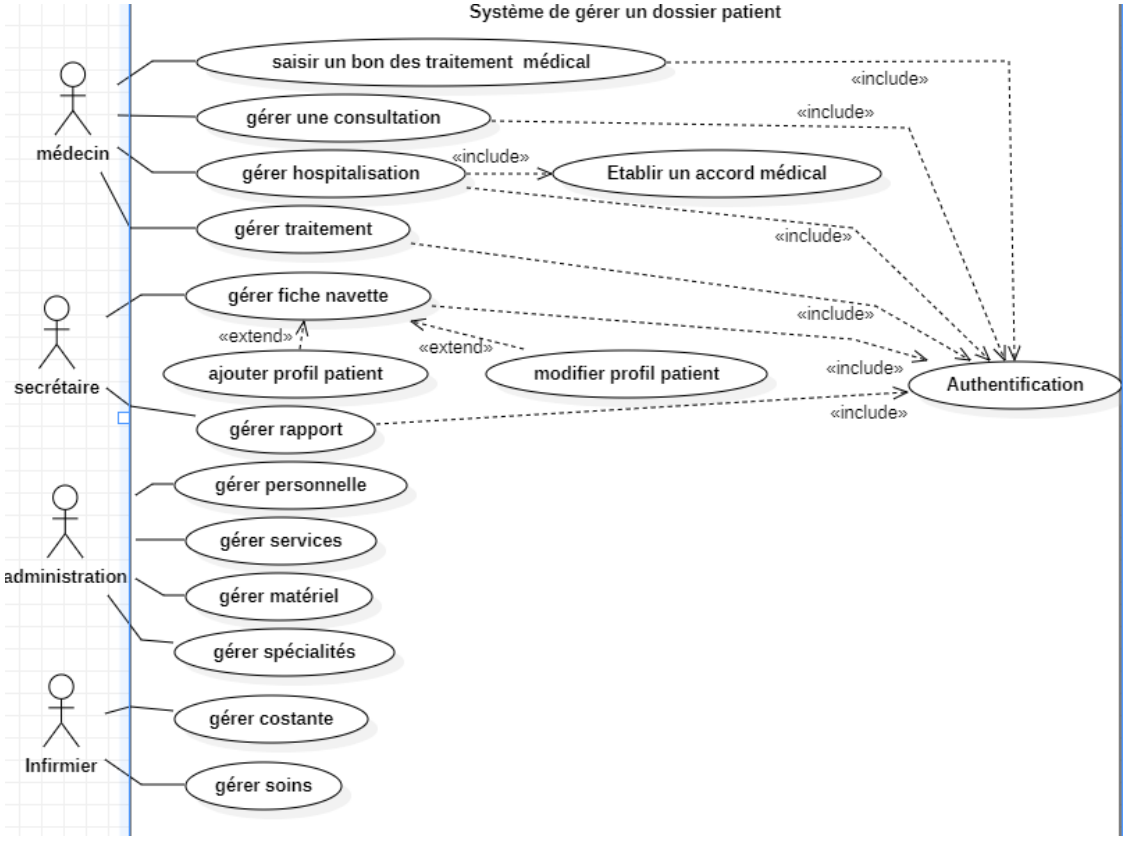


FIGURE 2.2 – diagramme de cas d'utilisation

2.5 Diagramme de classe Globale :

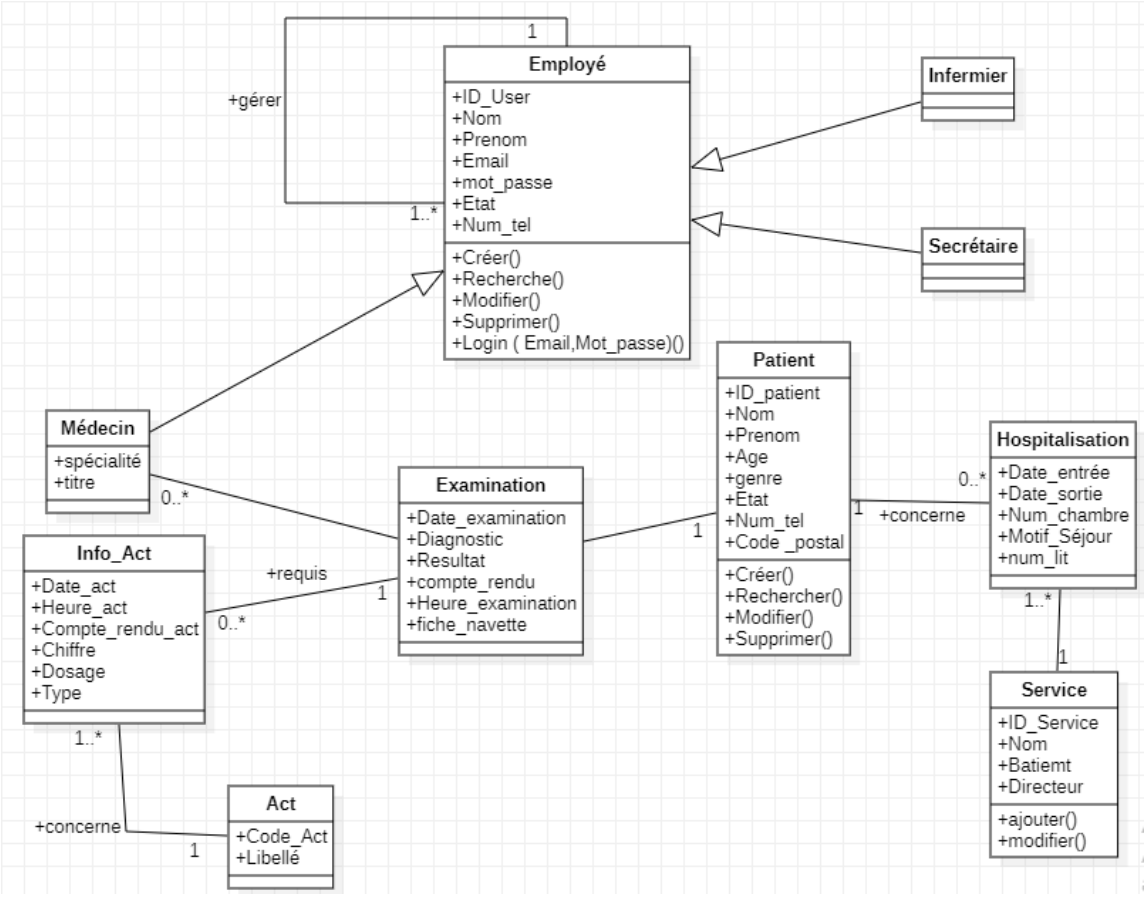


FIGURE 2.3 – diagramme de classe

2.5.1 Les règles de passage du MCD au MLD :

1- Une entité du MCD devient une relation, c'est à dire une table

2- Son identifiant devient la clé primaire de la relation.

3- Les autres propriétés deviennent les attributs de la relation

4- Une association de type 1 : N se traduit par la création d'une clé étrangère dans la relation correspondante à l'entité côté « 1 ». Cette clé étrangère référence la clé primaire de la relation correspondant à l'autre entité.

5- Une association de type N : N se traduit par la création d'une table dont la clé primaire est composée des clés étrangères référençant les relations correspondant aux entités liées par l'association. Les éventuelles propriétés de l'association deviennent des attributs de la relation.

Passage relationnel :

Employé (ID-Employé, Nom ,Prenom ,Email ,mot-passe ,Num-tel , spécialité , titre ,ID-

Superviseur,#ID-Emplyé)

Hospitalisation (ID-Hospitalisation, Date-Entrée , Date-sortie , Num-chambre , Motif-séjour , Num-Lit , #ID-Service , #ID-Patient)

Service (ID-Service, Nom, Batiment, Directeur)

Patient (ID-Patient, Nom, Prenom, Age, Genre, Num-tel, Code_ postal)

Examination (#ID- User, #ID-Patient, Date-examination, Diagnostic, Resultat, Compte-rendu Heure_examination, Demande, fiche navette)

Info_Act (Date_act, Heure _act, Compte _rendu_act, chiffre, dosage, type, #code_Act)

Act (code_Act, Libellé)

2.6 Modélisation du Processus métier :

2.6.1 Modélisation proposée :

Pour modéliser la variabilité dans notre projet, nous avons introduit une nouvelle notation dans notre diagramme BPMN. Nous avons utilisé des stéréotypes pour identifier les différentes variations et options dans notre processus. Le stéréotype "VP" a été ajouté au niveau des LANES pour indiquer une variation de processus (VP Service Médical). De plus, au niveau des tâches, nous avons introduit les stéréotypes "<obl>" pour les tâches obligatoires qui sont nécessaires pour chaque établissement de de santé que ce soit hôpital ou clinique (vérifier l'existence du patient) et "<opt>" pour les tâches optionnelles qui peuvent être varié d'un établissement à un autre (établir une fiche navette). Ainsi, en utilisant cette nouvelle notation, nous pouvons facilement visualiser et comprendre la variabilité de notre processus.

2.6.2 Processus métier "Gestion dossier patient" :

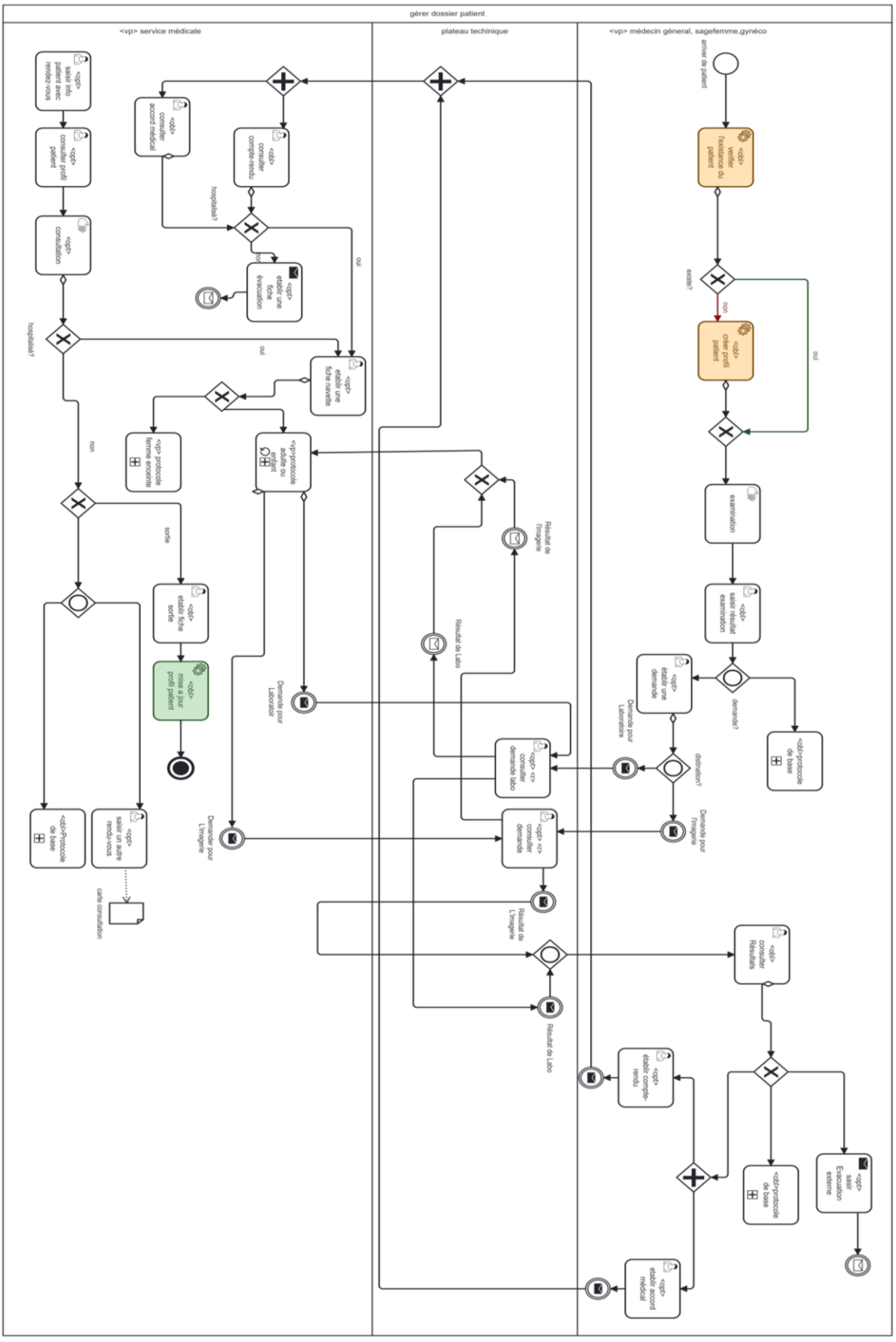


FIGURE 2.4 – diagramme BPMN

2.6.3 sous- processus de processus BPMN :

1- protocole de base :

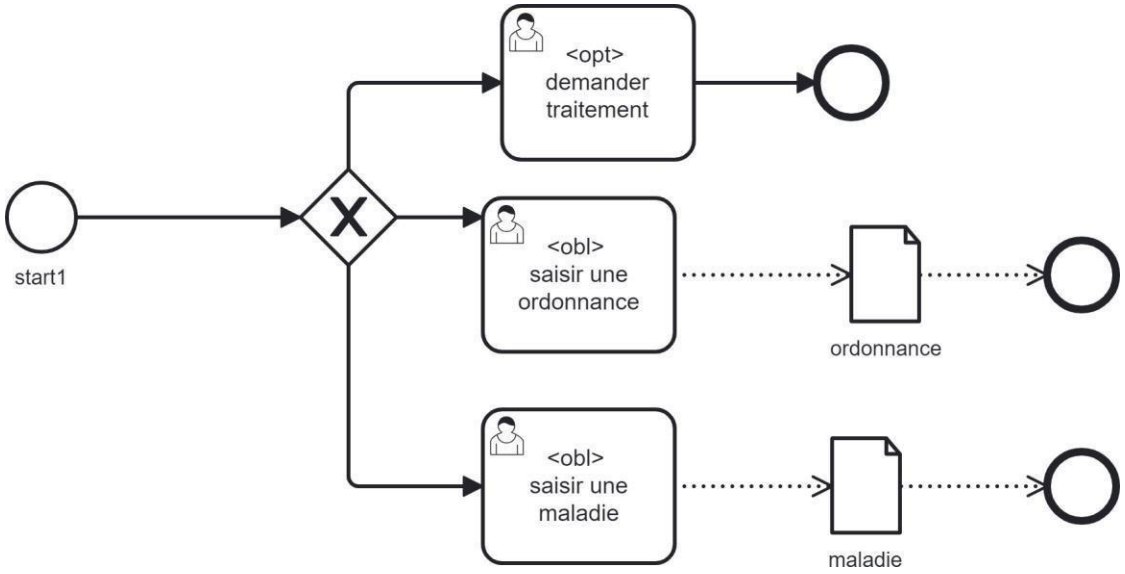


FIGURE 2.5 – protocole de base

2- protocole adulte ou enfant :

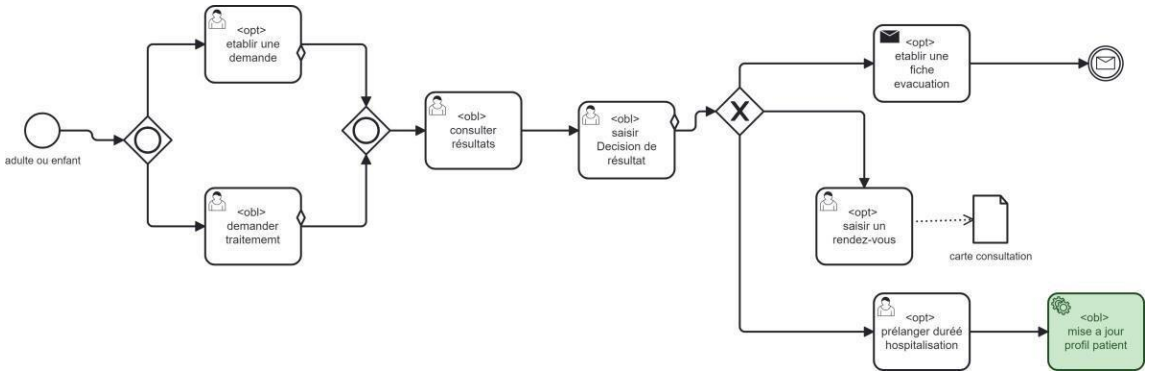


FIGURE 2.6 – protocole adulte ou enfant

3- protocole femme enceinte :

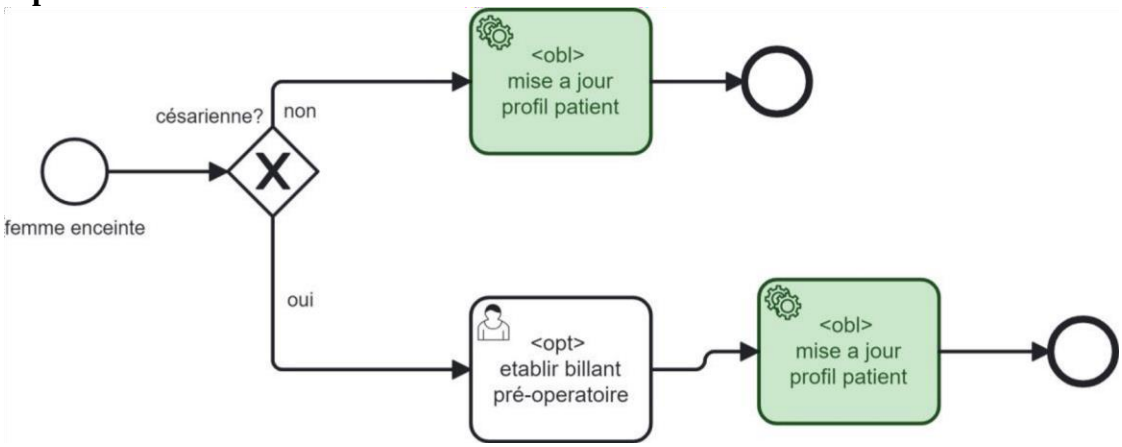


FIGURE 2.7 – protocole femme enceinte

2.6.4 Conclusion :

Dans ce chapitre, nous avons présenté l'analyse et la conception de notre système en utilisant UML et BPML. Dans le chapitre qui suit, nous allons présenter la phase réalisation de notre application web.

Chapitre 3

Implémentation

3.1 Introduction :

L'implémentation est l'étape décisive ou les idées abstraites prennent vie et deviennent des applications concrètes. Ce chapitre est consacré à la réalisation d'un produit fonctionnel en respectant la conception présenté dans le chapitre précédant, ce dernier est divisé en deux parties. La première partie qui parle sur les ressources logicielles utilisées et une deuxième partie sur l'explication des étapes qu'on a suivie pour la réalisation et la validation de notre solution.

3.2 Présentation des outils logiciels utilisés

Nous avons utilisé pour la réalisation de notre solution trois catégories de logiciels :

Eclipse :

Eclipse est un logiciel extensible, d'outils et de temps d'exécution qui a été initialement créée en tant qu'environnement de développement intégré (IDE) basé sur langage de programmation Java qui est un langage orientée objet pour coder des applications web.

MySQL :

MySQL est un système de gestion de bases de données relationnelles SQL open source développé et supporté par Oracle [17].

Camunda :

Camunda est une plateforme open-source pour la gestion des processus métiers, elle offre un ensemble des fonctionnalités aux organisations pour améliorer, exécuter, surveiller et optimiser leurs workflow, et dans la partie suivante, nous allons parler en détaille sur le fonctionnement de cette plateforme.

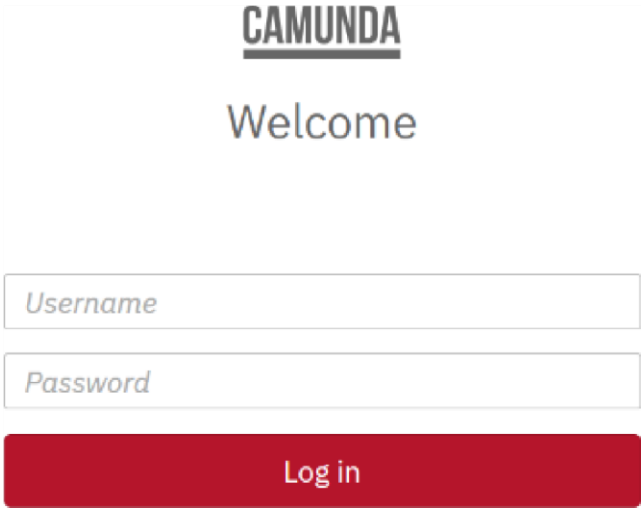


FIGURE 3.1 – Page d'accueil [8]

Apache Tomcat :

Apache Tomcat est un serveur Web largement utilise pour hébergé les applications Web java. Pour déployer et exécuter des applications basée sur Camanda, il est nécessaire d'utilise un serveur d'application java. Camunda peut être déployé sur diffèrent serveur d'applications tel que : WildFly, JBoss, Spring Boot, Tomcat… ect

Dans notre cas, nous avons choisis Tomcat pour facilite le déploiement, la gestion et exécution de Camunda de manière efficace.

FIGURE 3.2 – Apache Tomcat [8]

3.3 Architecture de la solution

Dans cette section, On va expliquer en détail l'architecture global de notre solution .Cette architecture va être basé sur la plateforme Camunda ou on va réaliser et tester notre processus métier.

3.3.1 L'architecture du Workflow Engine Camunda

La plateforme BPM Camunda offre plusieurs fonctionnalités pour la modélisation et optimisation des processus métiers même pour le développement des architectures orientées service.

Camanda fournir plusieurs application qui s'exécutent sur un réseau qui peut interagir avec le moteur de processus via un canal de communication à distance en utilisant l'API REST. Différents canaux de communication telque SOAP peut être implémentés par les utilisateurs. Cette plateforme peut être décomposé en deux partie : partie application et partie moteur comme montre la figure suivante :

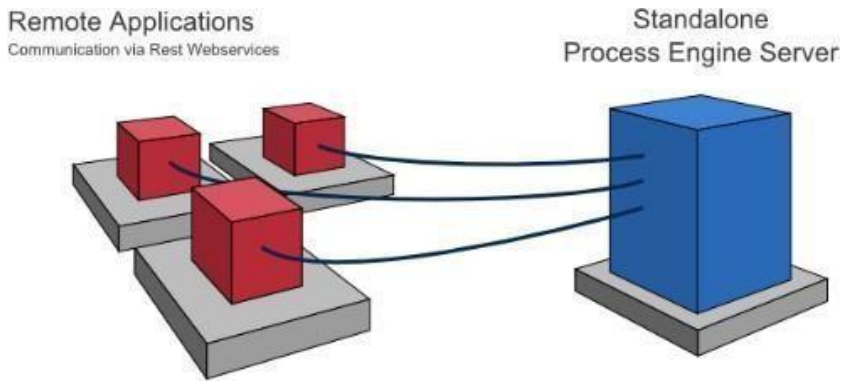


FIGURE 3.3 – l'architecture de déploiement de Camunda [7]

3.3.2 La modélisation des processus à travers les applications web et Camunda model

Camunda vient avec des applications web et du modeler pour faciliter le travail et la modélisation des processus métiers. Voici les différentes applications en détaille :

- **Camunda modeler** est un outil de modélisation pour les diagrammesBPMN 2.0 et CMNN 1.1 et pour les tables de décision DMN 1.3.
- **Camunda Tasklist** est une application web destiné pour les utilisateurs (participants du processus) pour interagir avec leurs tâches (user Tasks) et de naviguer vers des formulaires pour fournir des données.

- **Camunda Cockpit** est une application Web pour la surveillance des opérations qui vous permet de rechercher des instances de processus et d'inspecter leur état.

- **Camunda Admin** est une application Web qui permet de gérer les utilisateurs, les groupes et les autorisations.

3.3.3 Process Engine et Infrastructure

C'est la partie la plus importante, car elle repose sur une approche modulaire et distribuée. Elle contient le processus moteur qui est le responsable de l'exécution des processus BPMN 2.0, des cas CMMN 1.1 et des décisions DMN 1.3 .Elle contient un agent d'historisation qui va enregistrer et stocker les informations sur une base de données comme montre la figure suivante :

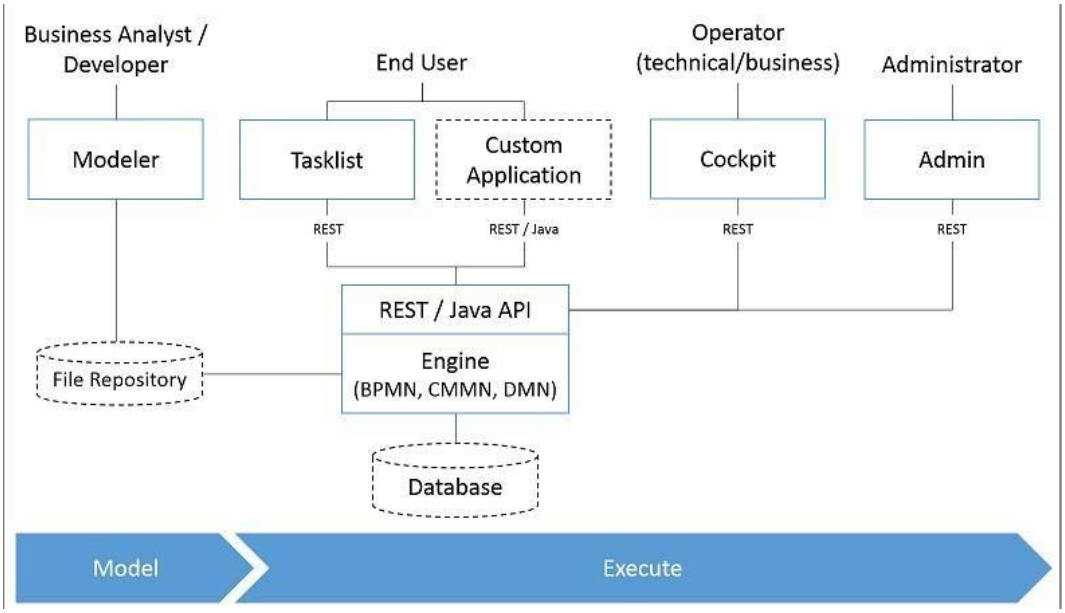


FIGURE 3.4 – Architecture générale de Camunda [8]

3.3.4 Base de données de Camunda

Le schéma de base de données du moteur de processus se compose de plusieurs tables. Tous les noms de tables commencent par ACT (Il convient de noter que l'utilisation du préfixe "ACT" dans les tables de base de données de Camunda est une convention interne de la plateforme pour organiser les données spécifiques à Camunda). La deuxième partie est une identification à deux caractères du cas d'utilisation de la table.

- **ACT-RE-* :** RE signifie référentiel. Les tableaux avec ce préfixe contiennent des informations « statiques » telles que les définitions de processus et les ressources de processus (images, règles, etc.).

- **ACT-RU-* :** RU signifie temps d'exécution. Ce sont les tables d'exécution qui contiennent les données d'exécution des instances de processus, des tâches utilisateur, des variables, des travaux, etc. Le moteur ne stocke les données d'exécution que pendant l'exécution de l'instance de processus et supprime les enregistrements lorsqu'une instance de processus se termine. Cela permet de garder les tables d'exécution petites et rapides.

- **ACT-ID-* :** ID signifie identité. Ces tables contiennent des informations d'identité telles que les utilisateurs, les groupes, etc.

- **ACT-HI-* :** HI signifie histoire. Ce sont les tables qui contiennent des données historiques telles que les instances de processus passées, les variables, les tâches, etc.

- **ACT-GE-* :** Données générales, qui sont utilisées dans divers casd'utilisation.

Les tables principales des moteurs de processus sont les entités des définitions de processus, des exécutions, des tâches, des variables et des abonnements aux événements. Leur relation est illustrée dans le modèle UML suivant :

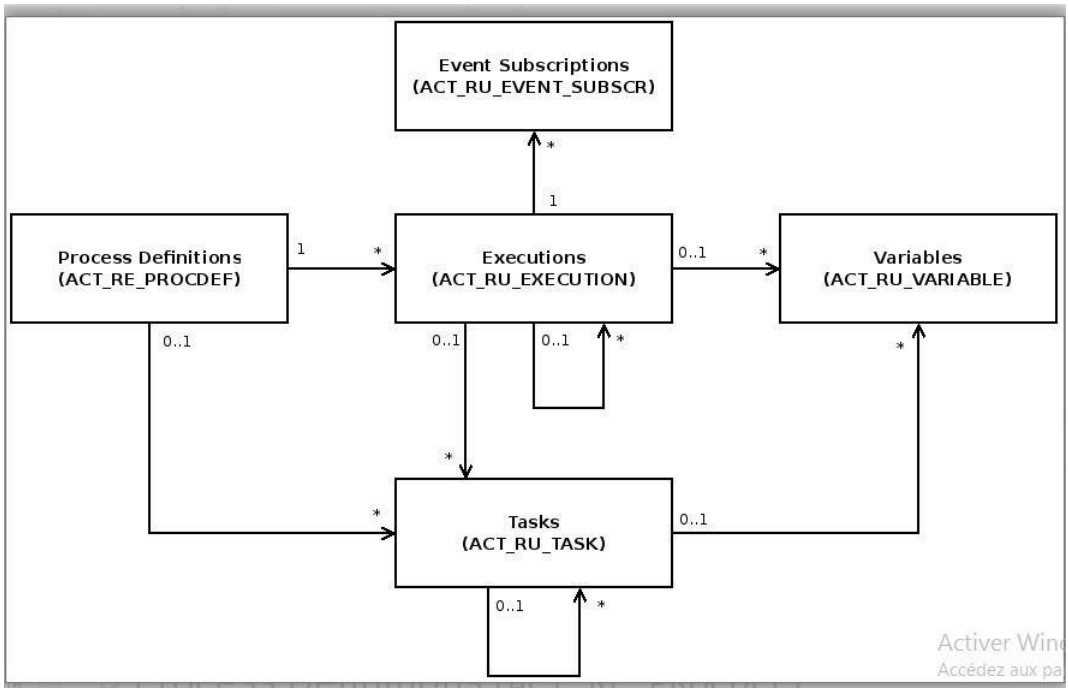


FIGURE 3.5 – Architecture des tables de base de données [8]

3.3.5 Exécution et déploiement de la solution

Après avoir expliqué les outils et l'architecture de développement de la solution, maintenant nous somme prêt d'entamer la phase de réalisation qui est considéré comme l'étape la plus importante dans notre projet de développement.

3.3.5.1 Installation et paramétrage de Camunda

Comme indiquer dans la section précédente, Camunda est composé de plusieurs applications qui tournent sur un seul conteneur qui est Tomcat dans la même machine.

Description de l'environnement :

-Java Runtime Envirenemment 20.0.1
-Tomcat 7.19.0

3.3.6 Déploiement de la solution :

On va d'abord installer Camunda (Prosess engine, Tasklist, Cockpit, Admin) sur le conteneur Tomcat et on va procéder tous les paramétrages nécessaire. L'un de plus important paramétrage est la configuration de la base de donnée .Camunda utilise H2 comme base de données intègre par défaut mais elle offre aussi la possibilité de se connecter a d'autre base de donnée relationnel, telles que MYSQL. Nous avons choisi MYSQL comme base de donnée pour notre solution et nous avons réussi a la connecter avec succès. Toutes les tables commençons par « ACT » ont été créé automatiquement avec succès de notre base de données comme montre la figure ci-dessus :

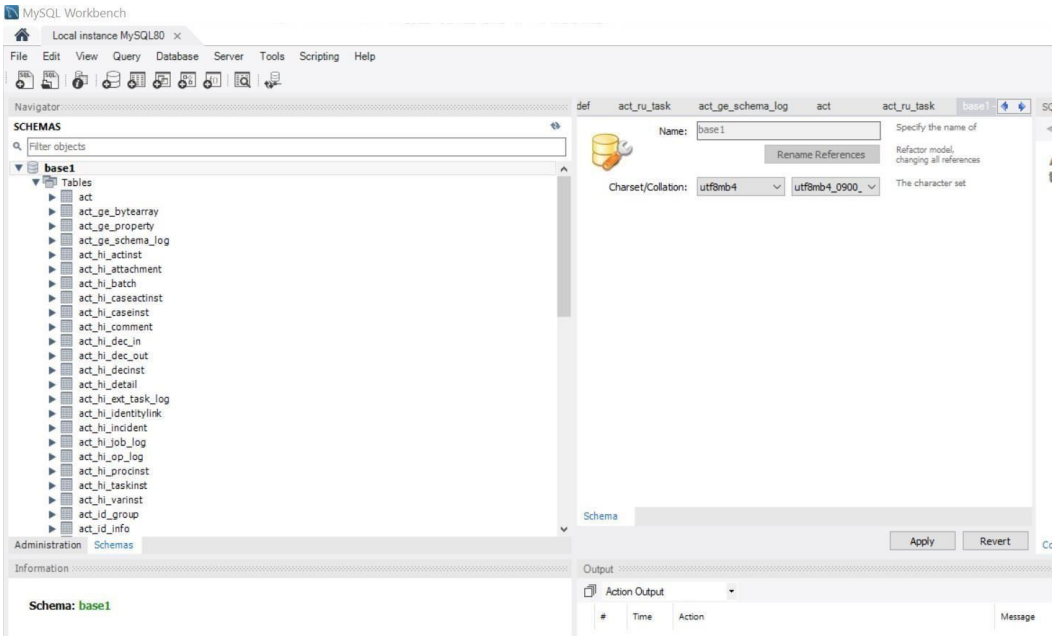


FIGURE 3.6 – Configuration de la base de données

Après la configuration de notre base de donnée et avant le déploiement de notre solution, il faut la rendre exécutable, C'est-à-dire pour chaque évènement ou Tâche (Utilisateur ou service ou ...) on va affecter le code Correspondant. La figure si dessous représente un exemple d'affectation d'un formulaire HTML à un Star Event sur le Camunda Modeler :

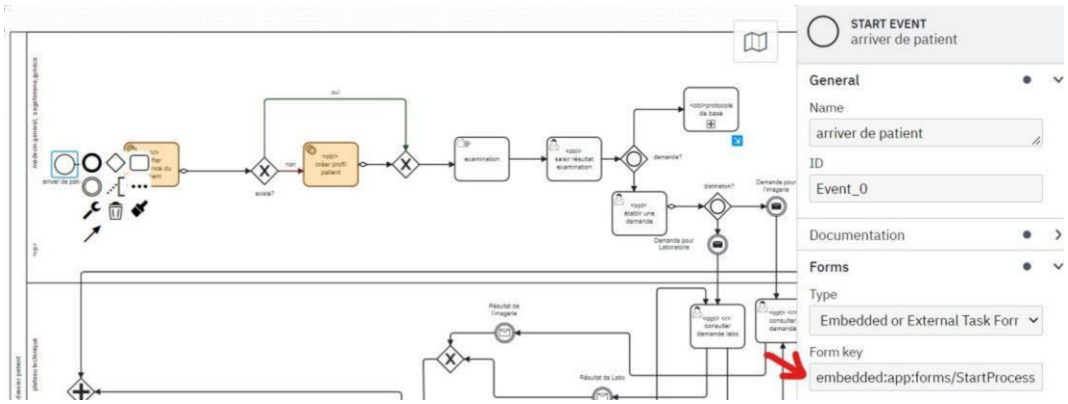


FIGURE 3.7 – Intégration de formulaire dans Camunda modeler

3.3.6.1 Déploiement de processus :

Les figures suivantes confirment que le processus était déployé correctement dans la console

Tomcat ensuite dans l'application Cockpit :

FIGURE 3.8 – Processus déployé dans la console de Tomcat

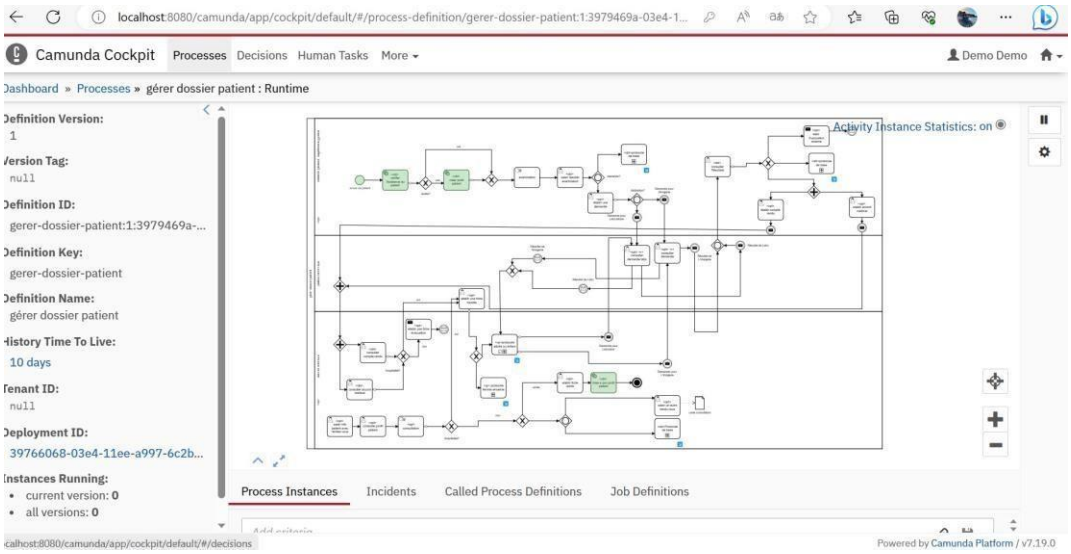


FIGURE 3.9 – Processus déployé dans l'application Cockpit

3.3.6.2 Configuration des utilisateurs, groupes et autorisations :

La figure suivant représente la liste des utilisateurs de notre système dans Camaunda :

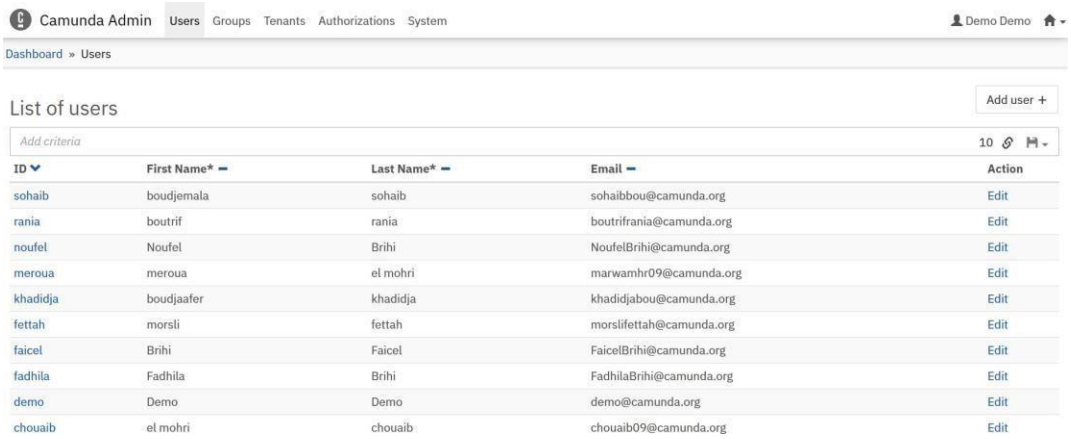


FIGURE 3.10 – Liste des utilisateurs

La figure suivant représente la liste des groupes de notre système dans Camaunda :

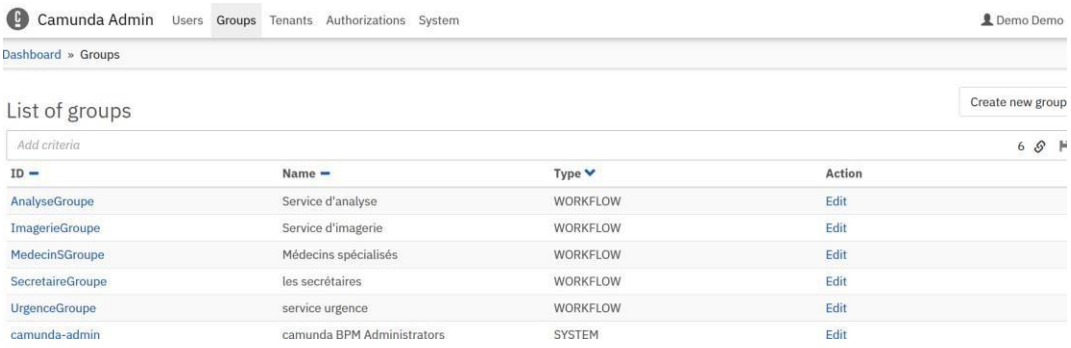


FIGURE 3.11 – Liste des groupes

On va dans cette partie configurer l'accès au diffèrent application :

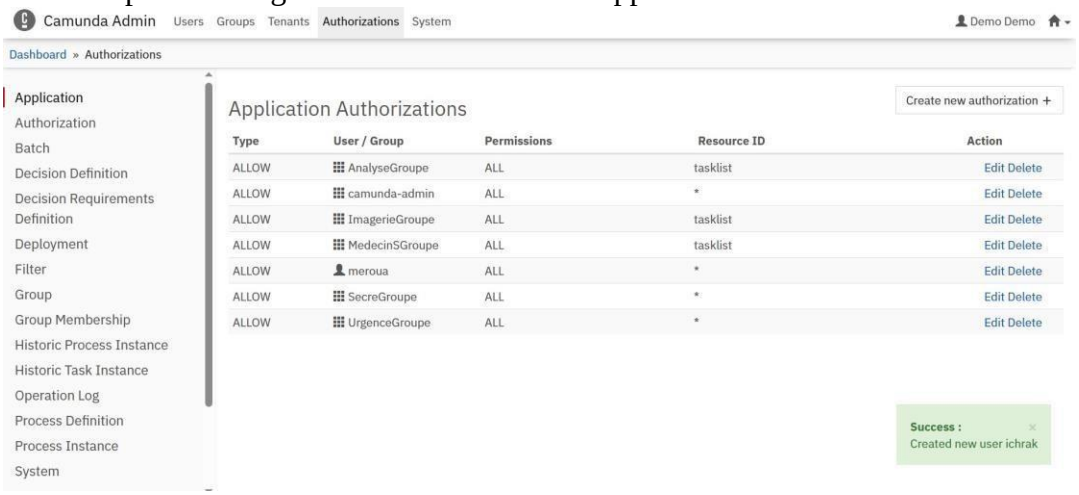


FIGURE 3.12 – Les droits de chaque Groupe utilisateur

3.3.7 Exécution de processus

Pour qu'un utilisateur puisse interagir avec notre processus, il est obligatoire de remplir cette formulaire pour il déclenche le début de processus :

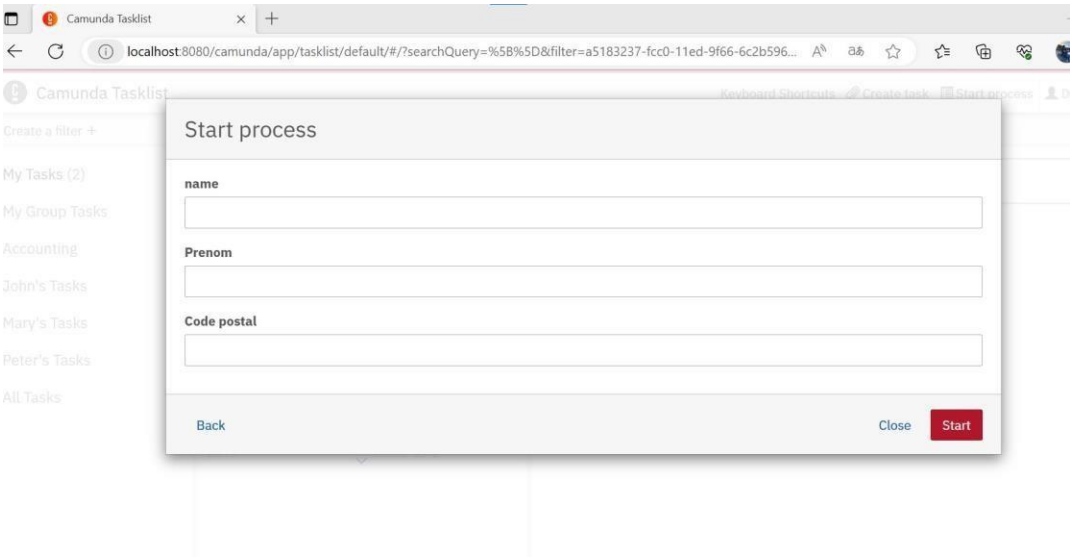


FIGURE 3.13 – Déclenchement de processus

3.3.7.1 Service Web :

Deux approches peuvent être utilisées pour la création des services web

Buttom-Up : création du logique métier dans un langage donné puis génération de WSDL.

Top-Buttom : définition d'un fichier WSDL puis génération d'un code vers un

Langage donné. Dans notre application, Nous avons suivi l'approche Buttom-Up sur le serveur Apache Tomcat.

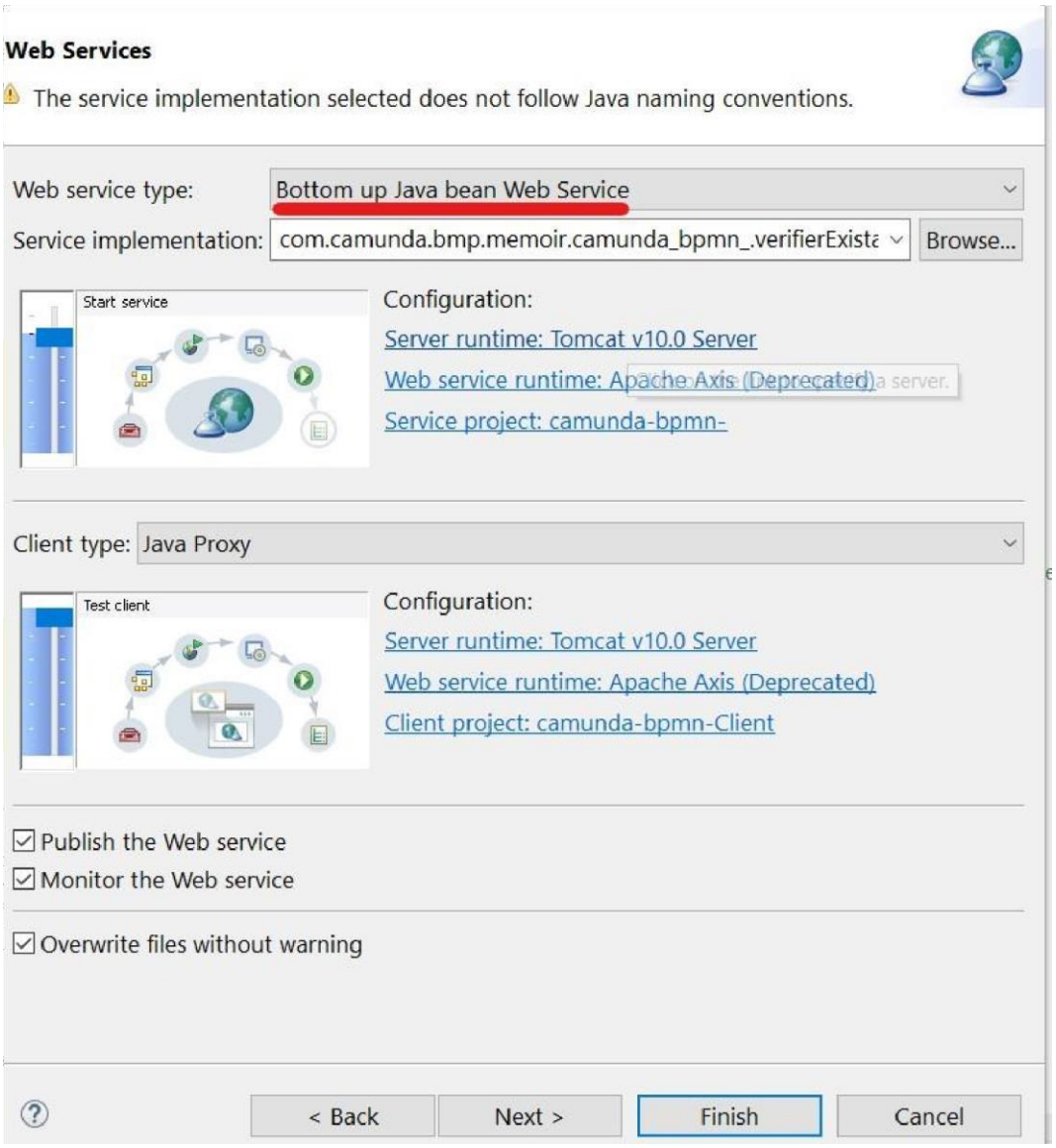


FIGURE 3.14 – L'approche Buttom-Up

Exemple d'un WSDL :

Un fichier WSDL contient une description de tout ce qui est nécessaire à l'appel d'un Service Web :

- L'URL et l'espace de noms du service.
- Le type de Service Web.
- La liste des fonctions disponibles.
- Les arguments de chaque fonction.
- Le type de données de chaque argument.
- La valeur de retour de chaque fonction et le type de données de chaque valeur de retour [13].

```
http://schemas.xmlsoap.org/wsdl/ (with catalog)
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://camunda.com" xmlns:apachesoap="http://xml.apache.org/xml-soap" xml
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="http://camunda.com" xmlns="http://www.w3.org/2001/
   <element name="main">
    <complexType>
     <sequence>
      <element maxOccurs="unbounded" name="args" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="mainResponse">
    <complexType/>
   </element>
   <element name="verifierPatient">
    <complexType>
     <sequence>
      <element name="nom" type="xsd:string"/>
      <element name="prenom" type="xsd:string"/>
      <element name="codePostal" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
   <element name="verifierPatientResponse">
    <complexType>
     <sequence>
      <element name="verifierPatientReturn" type="xsd:boolean"/>
     </sequence>
    </complexType>
   </element>
  </schema>
 </wsdl:types>

  <wsdl:message name="mainResponse">

     <wsdl:part element="impl:mainResponse" name="parameters">

     </wsdl:part>
```

FIGURE 3.15 – Exemple d'un fichier WSDL

3.4 Conclusion

Ce chapitre a présenté une approche importante et efficace pour le déploiement des processus métiers, il nous a permis de comprendre de maniéré détaillé Camunda, ces fonctionnalités et son rôle principale dans l'automatisation des processus métier et le domaine de l'architecture orienté service. Même on a examiné les étapes clé pour le déploiement et l'exécution de notre solution.

Conclusion général

Le projet qui nous a été confié consistait à réfléchir à une architecture orientée service dédiée au processus métier « le traitement d'un dossier patient dans un hôpital ».

Pour cela, nous avons effectué un travail indispensable de recherche et d'analyse sur l'architecture orientée services et l'automatisation de processus métier.

La présence d'un processus de développement formalisé, bien défini et bien géré est un facteur de réussite d'un projet. Pour cette raison nous avons suivi les fonctionnalités de la plateforme Camunda.

Au cours de notre projet nous avons développé un système qui permet aux utilisateurs de géré un dossier patient à partir d'un processus métier.

Ce projet très enrichissant, nous a donné l'occasion de nous plonger dans un problème concret, où nous avons pu mettre à profit les connaissances acquises durant notre cursus, et nous avons acquis de nouvelles compétences, plus particulièrement dans les architectures des systèmes d'information. Le début a été très difficile, ce qui est principalement dû à la multiplicité des concepts SOA, Variabilité, processus métier, Orchestration des services web avec Camunda et la notation BPMN.

Cependant, ces difficultés nous ont permis d'acquérir un esprit de synthèse, ce qui est loin d'être négligeable.

Le projet nous a de plus permis d'approfondir nos connaissances sur la programmation en JAVA, et le langage SQL, ainsi que l'acquisition de connaissance sur des techniques récentes de modélisation et exécution de processus métiers (Camunda). «

Néanmoins comme tout projet, le nôtre n'est pas exhaustif, il serait intéressant d'enrichir ce travail par : l'ajout d'une interface graphique (custom application) qui sera connecté avec la plateforme Camunda et plus précisément l'application web Tasklist pour bien démontrer et clarifier les fonctionnalités et les avantages de notre système.

Bibliographie

[1] Ana Moreira, A. R. e. J. A. (2005). Multi-dimensional separation of concerns in requirements engineering. Pages 285–296.

[2] Andreas Metzger, Klaus Pohl, P. H. P.-Y. S. e. G. S. (2007). Disambiguating the

Documentation of variability in software product lines : A separation of concerns, formalization and automated analysis. 1 :243–253.

[3] CARBONNEL, J. (Le 29 octobre 2018). L'analyse formelle de concepts : un cadre structurel pour l'étude de la variabilité de familles de logiciels. Master's thesis, L'UNIVERSITE DE MONTPELLIER.

[4] CARBONNEL, J. (September 2007). Evaluating à service-oriented architecture. Master's thesis, Software Engineering Institute.

[5] Creff, S. (le 09 décembre 2013). Une modélisation de la variabilité multidimensionnelle pour une évolution incrémentale des lignes de produits. Master's thesis, l'Université Européenne de Bretagne.

[6] Cumunda (2013a). https : //docs.camunda.org/manual/7.19/.

[7] Cumunda (2013b). "https : //docs.camunda.org/manual/7.19/introduction/architecture/".

[8] Cumunda (2020). https : //camunda.com/blog/2020/06/camunda-modeler-4-0-0-released/.

[9] Goiuria Sagardui, L. E. (2005). Variability driven quality evaluation in software product lines. Pages 243–252.

[10] M. Weske, A. H. T. H. (2003). Business process management : A survey. 2678.

[11] Fedjkhi, N. A. (2008). Une approche basée objectif pour la gestion des processus métier flexibles.

[12] Group), O. O. M. (Janvier 2011). Business process model and notation (bpmn), version 2.0.

[13] GUENDOUZ Amina, G. K. (2010/2011). Une architecture orientée service dédiée au processus métier : « le traitement d'une demande de raccordement en électricité/gaz. Master's thesis, l'université de SAAD DAHLBE.

[14] Halmans, G. and Pohl, K. (2004). Communicating the variability of a software Product family to customers. Informatik - forschung and entwicklung. 18 :113–131.

[15] Juanjuan Jiang, A. R. and Systä, T. (2005). Pattern-based variability management in web service development. Master's thesis, Tampere University of Technology, Institute of Software

Systems.
[16] K. C. Kang, S. G. Cohen, J. A. H. W. E. N. e. A. S. P. (novembre 1990). Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon University Software Engineering Institute.
[17] kinsta Ntd (2011). Kinsta.com.
[18] Klaus Pohl, G. B. e. F. J. v. d. L. (2005). Software Product Line Engineering : Foundations, Principles and Techniques. Springer NY, Inc.
[19] Maryam Razavian, R. K. (2008). Modeling variability in business process models using uml. Master's thesis, Department of Electrical and Computer Engineering, University of Tehran.
[20] McIlroy, M. D. (1968). Mass produced software components. Pages 138–155.
[21] Michael Kircher, C. S. e. I. G. (2006). Transitioning to a software product family approach - challenges and best practices. 163-171.
[22] Michiel Koning a, Chang-ai Sun b. M. S. a. P. A. a. (26 January 2008). Vxbpel : Supporting variability for web services in bpel. Master's thesis, a Department of Computer Science, University of Groningen.
[23] Mohammad Abu-Matar, Hassan Gomaa, M. K. and Elkhodary, A. (2010). Feature modeling for service variability management in service-oriented architectures. Master's thesis, George Mason University.
[24] Méghaichi Moufida, M. H. (2021/2022). Une approche outillée pour l'intégration des processus métier (bpmn) dans le développement des applications de l'internet des objets. Master's thesis, Université Mohamed Seddik Ben Yahia de Jijel.
[25] Parnas, D. (1979). On the design and development of program families. IEE Transactions on Software Engineering, SE 2 :1–9.
[26] Simon, A. (2014). Implémentation d'un éditeur bpmn au sein d'un outil de méta modélisation. Master's thesis, Université de Naumer.
[27] Soo Dong Kim, J. S. H. e. S. H. C. (2005). A theoretical foundation of variability in component-based development. Information and Software Technology, 47(10) :663–673.
[28] Tuan Nguyen, Alan Colman, M. A. T. and Han, J. (2015). Managing service variability : state of the art and open issues. Master's thesis, Faculty of Information and Communication Technology, Swinburne University of Technology, Melbourne, Australia.
[29] Vercruysse, T. (2015). Modélisation et instanciation de processus sur des solutions techniquement hétérogènes.
[30] (WMC), W. M. C. (1996). Workflow management coalition terminology and glossary (wfmc-tc-1011). Technical report, Workflow Management Coalition.

[31] Øystein Haugen, Birger Møller-Pedersen, J. O. G. K. O. e. A. S. (2008). Adding standardized variability to domain specific languages. Pages 139–148.