



REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie logiciel

Par :

ZERROUKI Djamel

Sur le thème

**Conception et mise en place d'un système Workflow
pour l'environnement cloud**

Soutenu publiquement le .. / 07 / 2019 à Tiaret devant le jury composé de :

Mr MEBAREK Bendaoud

MCA Université Ibn-khaldoun

Président

Mr AID Lahcene

MCB Université Ibn-khaldoun

Encadreur

Mr MEGHAZI Hadj Madani

MAA Université Ibn-khaldoun

Examinateur



REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie logiciel

Par :

ZERROUKI Djamel

Sur le thème

**Conception et mise en place d'un système Workflow
pour l'environnement cloud**

Soutenu publiquement le .. / 07 / 2019 à Tiaret devant le jury composé de :

Mr MEBAREK Bendaoud

MCA Université Ibn-khaldoun

Président

Mr AID Lahcene

MCB Université Ibn-khaldoun

Encadreur

Mr MEGHAZI Hadj Madani

MAA Université Ibn-khaldoun

Examinateur

اللهم لا علم لنا إلا ما علمتنا ، إِنَّكَ أَنْتَ الْعَلِيمُ الْحَكِيمُ ، اللَّهُمَّ عَلَمْنَا مَا يَنْفَعُنَا
، وَانْفَعْنَا بِمَا عَلِمْنَا ، وَزَدْنَا عِلْمَنَا ، وَأَرِنَا الْحَقَّ حَقًا ، وَارْزَقْنَا اتِّبَاعَهُ
وَأَرِنَا الْبَاطِلَ بَاطِلًا ، وَارْزَقْنَا اجْتِنَابَهُ ، وَاجْعَلْنَا مِنْ يَسْتَمْعُونَ الْقَوْلَ
فَيَتَبعُونَ أَحْسَنَهُ ، وَادْخُلْنَا بِرَحْمَتِكَ فِي عِبَادَكَ الصَّالِحِينَ.

Les batailles de la vie ne sont pas gagnées par les plus forts, ni par les plus rapides mais par ceux qui n'abandonnent jamais.

Dédicace

Je dédie ce modeste travail :

À ma mère :

Nulle dédicace ne saurait exprimer suffisamment ma gratitude, mon amour et mon profond respect à ma mère. Pour son soutien, son amour et son sacrifice. Sa présence et ses encouragements sont pour moi les piliers fondateurs de ce que je suis et de ce que je fais.

À mon père :

Puisque rien au monde ne pourrait compenser les sacrifices démesurés qu'il a déployés pour guider mes pas, et ses encouragements continus. Que mon père accepte à cette occasion, mes hommages comme gage de mon profond amour, et ma reconnaissance jamais interrompue.

À mes chers sœurs Habiba et Farah,

À mes très chères tantes, oncles, cousins et cousines,

À toute ma grande Famille,

À Mes Amis,

particulièrement : Ilyasse, Djamel, Sofiane, Walid, Abdelhak, Yacine, Mohamed, Houari, Khaled, Chahra .

Et à tous ceux qui ont été présent pour moi tout au long de mes études, et à toutes les personnes que j'apprécie et que je n'ai pas citées.

Remerciements

Avant de présenter notre modeste travail, il nous apparaît opportun de présenter d'abord nos remerciements :

*Avant tout, nous remercions **ALLAH**, le tout puissant qui nous a donnés à la fois, le courage et la patience pour bien mener ce travail.*

*Nous manifestons nos remerciements distingués à notre encadreur **M.LAHcen Aid** pour tous ses efforts fournis, son aide immense, ses conseils précieux et ses orientations pour atteindre nos objectifs.*

Nous adressons nos plus profondes reconnaissances à la communauté GitHub pour les bibliothèques open sources que nous avons utilisées, à StackOverFlow pour leurs réponses à nos questions techniques et à CodeAcademy, sans oublier tous les youtubers qui nous ont offert des formations gratuitement.

Et nous remercions les membres de jurys qui ont accepté d'évaluer ce travail de fin d'études.

Sans oublier, toute personne ayant contribué, de près ou de loin, à l'accomplissement de ce travail.

En espérant que ce modeste travail soit à la hauteur et reflète ce que nous avons pu acquérir pendant la durée du projet.

Résumé

La technologie de Workflow découle du besoin des entreprises d'améliorer leurs performances, notamment le contrôle des informations, une meilleure collaboration entre les parties prenantes et de meilleurs processus commerciaux et opérationnels.

L'objectif de notre projet est de concevoir et de réaliser un système Workflow administrative pour environnement Cloud.

Afin de répondre à cet objectif, nous avons développé un flux de travail administratif pour l'environnement Cloud, permettant l'exécution et le fonctionnement automatiques des processus organisationnels afin de formaliser, normaliser et accélérer le traitement des documents et autres tâches.

Pour réaliser ce travail, nous avons dû mener une étude théorique pour comprendre le domaine de Cloud et Workflow puis, utiliser les informations de cette étude pour compléter l'analyse, conception et l'implémentation de système.

Mots clés : Informatique en nuage (Cloud Computing), Flux de travail (Workflow), Processus Métier, Réseaux de Flux de travail, Spring Framework, Micro-Services.

Abstract

Workflow technology is driven by the need for businesses to improve their performance, including information control, better collaboration between stakeholders, and better business and operational processes.

The goal of our project is to design and build an administrative Workflow system for the Cloud environment.

In order to meet this objective, we have developed an administrative Workflow for the Cloud environment, enabling the automatic execution and operation of organizational processes to formalize, normalize and speed up the processing of documents and other tasks.

To perform this work, we had to conduct a theoretical study to understand the Cloud domain and Workflow then use the information from this study to complete the analysis, design and system implementation.

Keywords : Cloud Computing, Workflow, Business Process, Workflow Networks (WF-Net), Spring Framework, Micro-Services.

ملخص

تتبع تكنولوجيا سير العمل من حاجة الشركات لتحسين أدائها، بما في ذلك التحكم و السيطرة على المعلومات، وتحسين التعاون بين أصحاب المصلحة و ذلك بأفضل الأعمال التجارية والتشغيلية .

الهدف من مشروعنا هو تصميم وتنفيذ نظام سير عمل إداري تلقائي للبيئة السحابية لتحقيق هذا الهدف ، قمنا بتطوير هذا النظام و وضعه في بيئه سحابية ، مما يتتيح التنفيذ التلقائي و تشغيل العمليات التنظيمية لإضفاء الطابع الرسمي ، وتوحيد وتسريع معالجة الوثائق والمهام الأخرى .

لإكمال هذا العمل ، كان علينا إجراء دراسة نظرية لفهم مجال الحوسبة السحابية وسير العمل ، ثم استخدام معلومات هذه الدراسة لإجراء تحليل، تصميم وتنفيذ النظام.

الكلمات الرئيسية: حوصلة سحابية، سير العمل، العملية التجارية، شبكات سير العمل، إطار البرمجة الرابع، مايكرو الخدمات.

Table des matières

1 Concepts fondamentaux de cloud et de workflow	2
1.1 Introduction	2
1.2 Cloud computing	2
1.2.1 Concept du cloud computing	2
1.2.2 Caractéristiques principales du cloud computing	4
1.2.3 Technologies connexes	5
1.2.4 Modèles de service du cloud computing	7
1.2.5 Modèles de déploiement	9
1.2.6 Les principaux grands fournisseurs cloud pour 2019	10
1.3 Workflow	11
1.3.1 Introduction au Workflow	11
1.3.2 Origines	11
1.3.3 Définitions et terminologies	12
1.3.4 Définitions de base du Workflow	12
1.3.5 Description des Systèmes Workflow	14
1.4 Classification des systèmes Workflow	15
1.4.1 Processus collaboratifs	16
1.4.2 Workflow administratif	16
1.4.3 Workflow de production	17
1.4.4 Workflow adaptable ou Workflow ad hoc	17
1.4.5 Modèle de référence des systèmes Workflow	18
1.5 Intérêt du cloud pour les workflows	18
1.5.1 L'approvisionnement de ressources	19
1.5.2 L'allocation dynamique de ressources à la demande	19
1.5.3 L'élasticité	19
1.5.4 La garantie des QoS via des SLA	19
1.5.5 Le faible Coût d'exploitation	19
1.6 Modélisation de workflow avec les réseaux de pétri	20
1.7 les réseaux de Pétri	20
1.7.1 Qu'est-ce que les réseaux de Pétri	20
1.7.2 Représentation d'un réseau de Pétri	20
1.7.3 réseaux workflow (Wf-nets)	23
1.7.4 Constructions du routage dans un WF-net	23
1.8 Conclusion	25
2 Analyse et Conception	27
2.1 Unified Modeling Language (UML) :	27
2.1.1 Définition UML	27

2.1.2 Contenu UML	27
2.2 Identification des objectifs	28
2.2.1 Spécifications fonctionnelles	28
2.2.2 Spécifications techniques	29
2.2.3 Diagramme de cas d'utilisation	29
2.2.4 Documentation des cas d'utilisation fonctionnels	31
2.3 Représentation des informations	35
2.3.1 Diagramme de Classe	35
2.3.2 Diagramme d'activité	38
2.4 Conclusion	39
3 Réalisation	41
3.1 Introduction	41
3.2 Outils et Technologies utilisées pour le développement	41
3.2.1 Technologies web	41
3.2.2 Workflow :	42
3.2.3 Spring	42
3.2.4 Micro Service	44
3.2.5 Composantes majeures de Netflix	45
3.2.6 Spring Boot et Spring Cloud Netflix OSS – Micro Service Architecture	46
3.2.7 Maven	46
3.3 Création d'un service cloud par spring et micro service :	49
3.3.1 Outils et Versions	49
3.3.2 Création des Microservices	50
3.3.3 Consulter les services via le proxy	55
3.4 Description de l'architecture de notre système	56
3.4.1 Obstacles	57
3.5 Présentation de système	57
3.5.1 Interface de l'administrateur	57
3.5.2 l'interfaces de l'Utilisateur	62
3.6 Conclusion	63
A Service 1	i
A.1 classe User implémenté en java	i
A.2 créer l'interface UserRepository	ii
A.3 classe DummyDataCLR en java	ii
B ConfigService	iii
B.1 classe ConfigServiceApplication en java	iii
B.2 json	iii
B.3 json	iv
B.4 classe userRestService en java	iv
C Microservice DiscoveryService	vi
D Microservice ProxyService	vii
E L'organigramme de la CNR	viii

Table des figures

1.1	Prévisions de la taille du marché du cloud computing public (RIED et KISKER 2011)	3
1.2	Convergence des différentes avancées menant à l'avènement du cloud computing.(COMPUTING 2019)	5
1.3	Une architecture en couches de la technologie de virtualisation.	7
1.4	Les services XaaS du cloud computing	8
1.5	Modèles de déploiement du cloud computing	10
1.6	Les systèmes de gestion de Workflow dans une perspective historique	12
1.7	Environnement du système Workflow (ZACHAREWICZ 2006)	12
1.8	Dimensions des systèmes Workflow	13
1.9	Caractéristiques du système Workflow	14
1.10	Différentes classes des systèmes Workflow	16
1.11	Modèle de référence des systèmes de gestion de workflow (WfMC, 95).	18
1.12	Réseau de Pétri simple	21
1.13	Réseau de Pétri marqué	22
1.15	Procédure modélisée avec le WF-net	23
1.16	Routage séquentiel.	24
1.17	Routage parallèle.	24
1.18	Routage conditionnel non-déterministe	24
1.19	Choix explicite entre B et C à base de l'attribut x.	25
1.20	Itération : B peut être exécutée plusieurs fois	25
2.1	Diagramme de Cas d'utilisation	30
2.2	Diagramme de Classe	35
2.3	Diagramme d'activité qui montre interaction des utilisateurs en fonction de leurs rôles dans le système en général.	39
3.1	Responsive design	42
3.2	Architecture Spring cloud et Netflix par des microservices	50
3.3	Microservice Service 1	50
3.4	Microservice ConfigService	52
3.5	Microservice DiscoveryService	54
3.6	Microservice ProxyService	55
3.7	Diagramme séquence représenté comment Consulter les services via le proxy	56
3.8	Notre architecture de système sur l'environnement Spring cloud et micro-service	57
3.9	Authentification au système	58
3.10	Un workflow pour un service de la Caisse Nationale Des Retraites (CNR)	58

3.11 Gestion des Tâches	59
3.12 Consultation Historique par Tâche	59
3.13 Gestion des Sous-Directions et des Services	60
3.14 Gestion des Utilisateurs	60
3.15 Consultation des Dossiers	61
3.16 Consultation Historique par dossier	61
3.17 La page d'accueil	62
3.18 Ajouter un dossier	62
3.19 Liste des dossier	63
3.20 Gérer les bordereaux	63
E.1 L'organigramme de la CNR	viii

Liste des tableaux

2.1	Les acteurs de système	29
2.2	La classe Dossier	36
2.3	La classe Service	36
2.4	La classe MyTask	36
2.5	La classe Utilisateur	36
2.6	La classe NextTask	37
2.7	La classe Sous-Direction	37
2.8	La classe Historique	37
2.9	La classe Role	37
2.10	La classe ListDossier	37
2.11	La classe Bordereau	38
3.1	Architecture des microservices via les composants Netflix	45

Liste Des Abréviations

ALS Service Level Agreement .

API Application Programming Interface

AWS Amazon Web Services

BPaaS Business Process as-a-Service

BPR Business Process Reengineering

CNR Caisse Nationale Des Retraites

CPU Central Processing Unit

EC2 Elastic Compute cloud.

GAE Google App Engine

HaaS Hardware as-a-Service

IaaS Infrastructure as-a-Service

IBM International Business Machines .

NIST National Institute of Standards and Technology

OSS Open Source Software

PaaS Platform as-a-Service

QoS Quality Of Service

REST Representational State Transfer

SaaS Software as-a-Service

SGBD Système de Gestion de Bases de Données

UIMS User Interface Management Systems

UML Unified Modeling Language

UP Unified Process

VMM Virtual Machine Monitor

VPN Virtual Private Network

WAPI Workflow Application Programming interface

WFMS WorkFlow Management System

WfMC Workflow Management Coalition

WFMC-TC Workflow Management Coalition Terminology and Glossary

XaaS X as-a-Service

Introduction Générale

Contexte et Motivation

Le Cloud computing présente une technologie prometteuse qui facilite l'exécution des applications. Il fournit des services flexibles et évolutifs, à la demande des utilisateurs, via un modèle de paiement à l'usage. Généralement, il peut fournir quatre types de services : SaaS (Software as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service) et HaaS (Hardware as a Service). Ces services sont offerts avec différents niveaux de qualité de service afin de répondre aux besoins spécifiques de différents utilisateurs. Bien que de nombreux services de Cloud computing ont des fonctionnalités similaires (par exemple des services de calculs, services de stockages, services réseaux, etc.),

Parmi les applications, qui peuvent être utilisées en tant que service Cloud, nous trouvons celles de flux de travail (Workflow) qui permettent l'exécution et le fonctionnement automatisés des processus d'organisation pour formaliser, normaliser et accélérer le traitement des documents et autres tâches.

Problématique

Les entreprises modernes doivent faire preuve de capacités d'adaptation très importantes et d'une pratique du management et de l'organisation aussi souple que durable dans le temps. La technologie du Workflow est née du besoin qu'ont les entreprises d'améliorer leurs performances, notamment grâce à une maîtrise de l'information, une meilleure coopération des acteurs et une optimisation des processus opérationnels et métiers. Les solutions Workflow n'arrivent pas encore à démarrer en Algérie. C'est la raison pour laquelle nous proposons un Workflow pour le traitement des dossiers administratifs notamment dans un environnement Cloud.

Objectifs

1. Conception et implémentation d'un service Cloud.
2. Développement d'un service Workflow administratif pour l'environnement Cloud notamment pour réduire le temps de traitement des dossiers.
3. Développement d'un système générique qui peut être exécuté par plusieurs organisations.

Organisation du mémoire

Nous avons organisé notre mémoire comme suit :

Chapitre 1 : Concepts fondamentaux de Cloud et de Workflow

Ce chapitre vise à présenter les concepts fondamentaux de Cloud computing et de Workflow

Chapitre 2 : Analyse et Conception

A travers ce chapitre, nous avons clarifié et détaillé les besoins à travers différents diagrammes.

Chapitre 3 : Réalisation

Ce chapitre, décrit le choix des techniques et les outils de développement de notre solution, l'architecture technique, l'architecture du code et la configuration de application

Chapitre 1

Concepts fondamentaux de
Cloud et de Workflow

Chapitre 1

Concepts fondamentaux de cloud et de workflow

1.1 Introduction

Le cloud computing, traduit le plus souvent en français par " informatique dans les nuages ", " informatique dématérialisée " ou encore " infonuagique ", est un domaine qui regroupe un ensemble de techniques et de pratiques consistant à accéder, en libre-service, à du matériel ou à des logiciels informatiques, à travers une infrastructure réseau (Internet). Ce concept rend possible la distribution des ressources informatiques sous forme de services pour lesquels l'utilisateur paie uniquement pour ce qu'il utilise. Ces services peuvent être utilisés pour exécuter des applications scientifiques et commerciales, souvent modélisées sous forme de workflows.

Ce chapitre présente une introduction au cloud computing et au workflow, nécessaire pour la compréhension générale de ce mémoire.

1.2 Cloud computing

1.2.1 Concept du cloud computing

L'idée principale du cloud est apparue dans les années 60, où le professeur John McCarthy avait imaginé que les ressources informatiques seront fournies comme des services d'utilité publique (GARFINKEL et ABELSON 1999). C'est ensuite, vers la fin des années 90, que ce concept a pris de l'importance avec l'avènement du grid computing (FOSTER et KESSELMAN 1999). Le terme cloud est une métaphore exprimant la similarité avec le réseau électrique, dans lequel l'électricité est produite dans de grandes centrales, puis disséminée à travers un réseau jusqu'aux utilisateurs finaux. Ici, les grandes centrales sont les Datacenter, le réseau est le plus souvent celui d'Internet et l'électricité correspond aux ressources informatiques. Le cloud computing n'est véritablement apparu qu'au cours de l'année 2006 (VOUK 2008) avec l'apparition d'Amazon Elastic Compute cloud (EC2). C'est en 2009 que la réelle explosion du cloud survint avec l'arrivée sur le marché de sociétés comme Google (Google App Engine), Microsoft (Microsoft Azure), IBM (IBM Smart Business Service), Sun (Sun cloud) et Canonical Ltd (Ubuntu Enterprise cloud). Sun (Sun cloud) et Canonical Ltd (Ubuntu Enterprise cloud). D'après une étude menée par Forrester (RIED et KISKER 2011), le marché du cloud computing s'élevait à environ 5,5 milliards de

dollars en 2008, il devrait atteindre plus de 150 milliards d'ici 2020, comme l'illustre la figure 1.1.

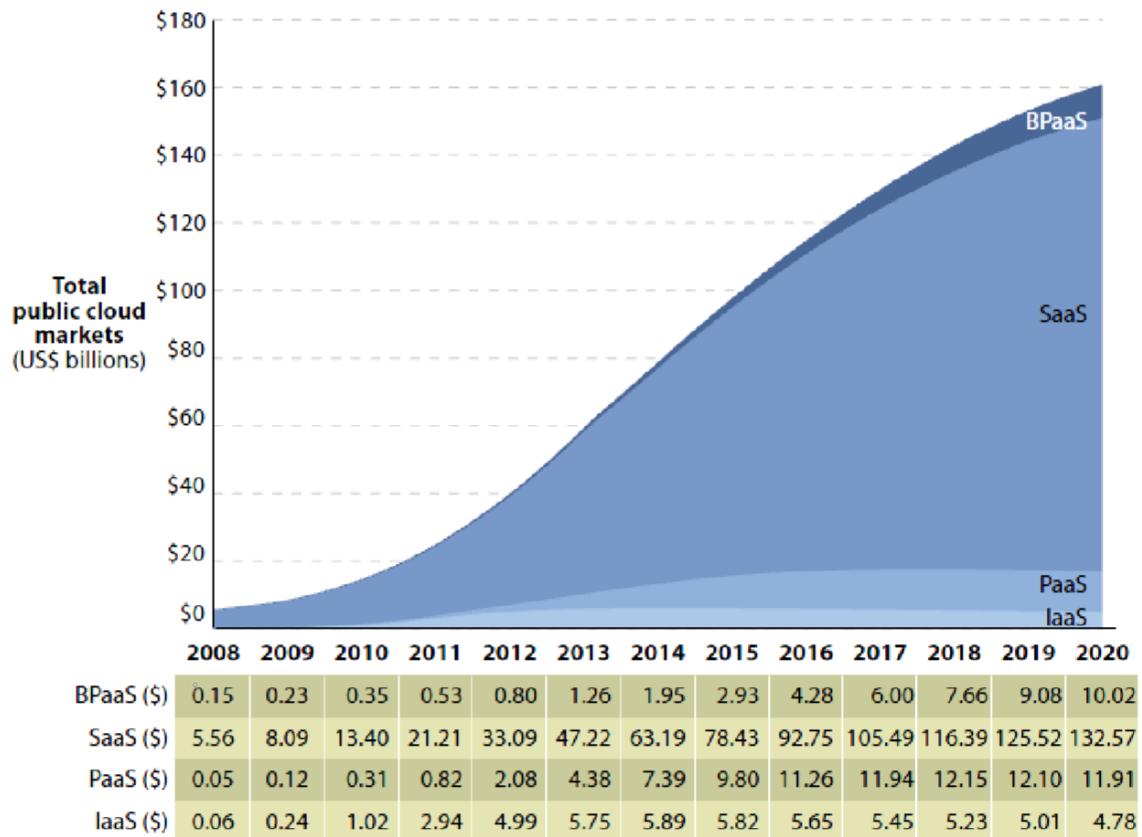


FIGURE 1.1 – Prévisions de la taille du marché du cloud computing public (RIED et KISKER 2011).

1.2.1.1 Vers une définition du cloud computing

Beaucoup de chercheurs ont tenté de définir le cloud computing (GEELAN 2019). La plupart des définitions attribuées à ce concept semblent se concentrer seulement sur certains aspects technologiques. L'absence d'une définition standard a généré non seulement des exagérations du marché, mais aussi des confusions. Pour cette raison, il y a eu récemment des travaux sur la normalisation de la définition du cloud computing, à l'exemple de Vaquero et coll (Vaquero, 2009) qui ont comparé plus de 20 définitions différentes et ont proposé une définition globale. En guise de synthèse des différentes propositions données dans la littérature, nous introduisons une définition mixte, qui correspond aux différents types de cloud considérés dans les travaux réalisés dans ce mémoire.

Définition 1.2.1. le cloud est comme un modèle informatique qui permet d'accéder, d'une façon transparente et à la demande, à un pool de ressources hétérogènes physiques ou virtualisées (serveurs, stockage, applications et services) à travers le réseau. Ces ressources sont délivrées sous forme de services reconfigurables et élastiques, à base d'un modèle de paiement à l'usage, dont les garanties sont offertes par le fournisseur via des contrats de niveau de Service Level Agreement (ALS).

1.2.2 Caractéristiques principales du cloud computing

1.2.2.1 Accès en libre-service à la demande

Le cloud computing offre des ressources et services aux utilisateurs à la demande. Les services sont fournis de façon automatique, sans nécessiter d'interaction humaine (MELL 2011).

1.2.2.2 Accès réseau universel

Les services de cloud computing sont facilement accessibles au travers du réseau, par le biais de mécanismes standard, qui permettent une utilisation depuis de multiples types de terminaux (par exemple, les ordinateur portables, tablettes, smartphones) (MELL 2011).

1.2.2.3 Mutualisation de ressources (Pooling)

Les ressources du cloud peuvent être regroupées pour servir des utilisateurs multiples, pour lesquels des ressources physiques et virtuelles sont automatiquement attribuées (MELL 2011). En général, les utilisateurs n'ont aucun contrôle ou connaissance sur l'emplacement exact des ressources fournies. Toutefois, ils peuvent imposer de spécifier l'emplacement à un niveau d'abstraction plus haut.

1.2.2.4 Scalabilité et élasticité

Des ressources supplémentaires peuvent être automatiquement mises à disposition des utilisateurs en cas d'accroissement de la demande (en réponse à l'augmentation des charges des applications) (GEELAN 2019), et peuvent être libérées lorsqu'elles ne sont plus nécessaires. L'utilisateur a l'illusion d'avoir accès à des ressources illimitées à n'importe quel moment, bien que le fournisseur en définisse généralement un seuil (par exemple : 20 instances par zone est le maximum possible pour Amazon EC2).

1.2.2.5 Autonome

Le cloud computing est un système autonome et géré de façon transparente pour les utilisateurs. Le matériel, le logiciel et les données au sein du cloud peuvent être automatiquement reconfigurés, orchestrés et consolidés en une seule image qui sera fournie à l'utilisateur (WANG 2008).

1.2.2.6 Paiement à l'usage

La consommation des ressources dans le cloud s'adapte au plus près aux besoins de l'utilisateur. Le fournisseur est capable de mesurer de façon précise la consommation (en durée et en quantité) des différents services (Central Processing Unit (CPU), stockage, bande passante,...); cela lui permettra de facturer l'utilisateur selon sa réelle consommation (Armbrust, 2009).

1.2.2.7 Fiabilité et tolérance aux pannes

Les environnements cloud tirent parti de la redondance intégrée du grand nombre de serveurs qui les composent en permettant des niveaux élevés de disponibilité et de fiabilité pour les applications qui peuvent en bénéficier (BUYYA et VENUGOPAL 2004).

1.2.2.8 Garantie Quality Of Service (QoS)

Les environnements de cloud peuvent garantir QoS la qualité de service pour les utilisateurs, par exemple, la performance du matériel, comme la bande passante du processeur et la taille de la mémoire (WANG 2008).

1.2.2.9 Basé-SLA

Les clouds sont gérés dynamiquement en fonction des contrats d'accord de niveau de service (SLA) (BUYYA et VENUGOPAL 2004) entre le fournisseur et l'utilisateur. Le SLA définit des politiques, telles que les paramètres de livraison, les niveaux de disponibilité, la maintenabilité, la performance, l'exploitation, ou autres attributs du service, comme la facturation, et même des sanctions en cas de violation du contrat. Le SLA permet de rassurer les utilisateurs dans leur idée de déplacer leurs activités vers le cloud, en fournissant des garanties de QoS. Après avoir présenté les caractéristiques essentielles d'un service cloud, nous présentons, brièvement, dans la section suivante, quelques technologies connexes aux clouds.

1.2.3 Technologies connexes

Le cloud computing utilise des technologies telles que la virtualisation, l'architecture orientée services et les services web. Le cloud est souvent confondu avec plusieurs paradigmes informatiques, tels que le Grid computing, l'Utility computing et l'Autonomic computing, dont chacun partage certains aspects avec le cloud computing. La figure 1.2 montre la convergence des technologies qui ont contribué à l'avènement du cloud computing.

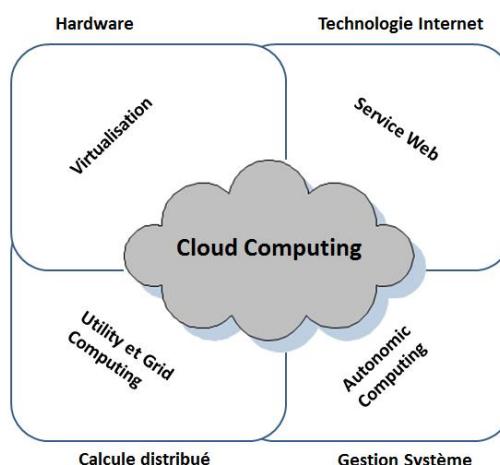


FIGURE 1.2 – Convergence des différentes avancées menant à l'avènement du cloud computing.(COMPUTING 2019)

1.2.3.1 Grid computing :

Le grid computing est un paradigme de calcul réparti qui coordonne des ressources autonomes et géographiquement distribuées pour atteindre un objectif de calcul commun. Le grid est basé sur le partage dynamique des ressources entre des participants, des organisations et des entreprises dans le but de pouvoir les mutualiser, et faire ainsi exécuter des applications scientifiques, qui sont généralement des calculs intensifs ou des traitements de très gros volumes de données. Le cloud computing est similaire au grid computing :les deux adoptent le concept d'offrir les ressources sous forme de services. Toutefois, ils sont différents dans leur finalité : tandis que la technologie des grilles vise essentiellement à permettre à des groupes différents de partager l'accès en libre service à leurs ressources, le cloud a pour objectif de fournir aux utilisateurs des services « à la demande », sur le principe du « paiement à l'usage ». De plus le cloud exploite les technologies de virtualisation à plusieurs niveaux (matériel et plateforme d'application) pour réaliser le partage et l'approvisionnement dynamique des ressources (MERIEM 2012).

1.2.3.2 Utility computing (informatique utilitaire) :

L'utility computing est simplement une délocalisation d'un système de calcul ou de stockage. Il présente un modèle d'affectation des ressources à la demande et facturation des utilisateurs en fonction de leur usage. Le cloud computing peut être perçu comme une réalisation de l'informatique utilitaire. Il adopte un système de tarification basé sur l'utilité, entièrement pour des raisons économiques. Avec l'approvisionnement des ressources à la demande et le paiement à l'usage, les fournisseurs de services peuvent maximiser l'utilisation des ressources et minimiser leurs coûts d'exploitation.

1.2.3.3 L'autonomic computing :

L'autonomic computing vise à construire des systèmes informatiques capables de s'auto-administrer en s'adaptant à des changements internes et externes sans intervention humaine. Le but de l'informatique autonome est de surmonter la complexité de la gestion des systèmes informatiques d'aujourd'hui. Bien que le cloud computing présente certaines caractéristiques autonomes, telles que l'approvisionnement automatique des ressources, son objectif est de diminuer le coût des ressources, plutôt que de réduire la complexité du système (YASSA 2014).

1.2.3.4 Virtualisation :

La virtualisation est une technologie qui fait abstraction des détails du matériel physique et fournit des ressources virtualisées pour les applications de haut niveau. En effet, la virtualisation regroupe l'ensemble des techniques matérielles ou logicielles permettant de faire fonctionner, sur une seule machine physique, plusieurs configurations informatiques (systèmes d'exploitation,...), de sorte à former plusieurs machines virtuelles, qui reproduisent le comportement des machines physiques. La virtualisation constitue le socle du cloud computing, car elle offre la possibilité de mettre en commun des ressources informatiques à partir de clusters de serveurs, et ainsi d'affecter ou réaffecter dynamiquement des machines virtuelles aux applications à la demande. La figure 1.3. montre l'architecture en couches de la technologie

de virtualisation. Le Virtual Machine Monitor (VMM) en français moniteur de machine virtuelle , également appelé hyperviseur, partitionne la ressource physique du serveur physique sous-jacente en plusieurs machines virtuelles différentes, chacune fonctionnant sous un système d'exploitation distinct et une pile de logiciels utilisateur (BOUCHHIMA 2006).

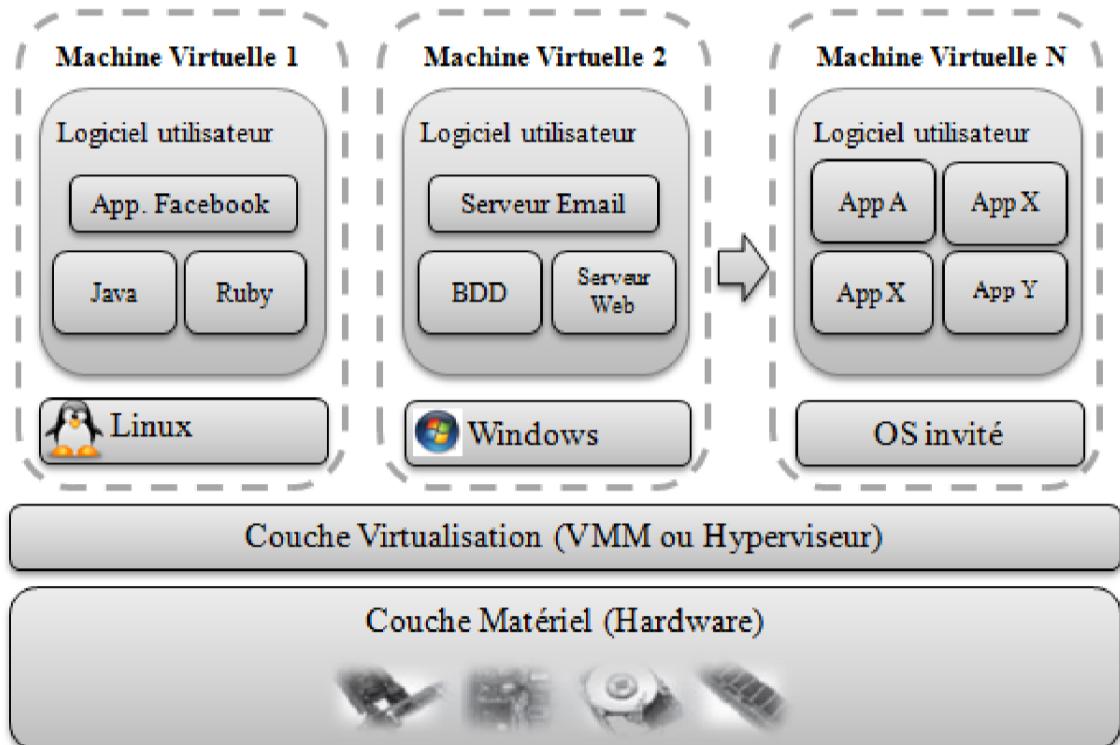


FIGURE 1.3 – Une architecture en couches de la technologie de virtualisation.

1.2.4 Modèles de service du cloud computing

X as-a-Service (XaaS) représente la base du paradigme du cloud computing, où X représente un service tel qu'un logiciel, une plateforme, une infrastructure, un Business Process Business Process as-a-Service (BPaaS), etc. Nous présentons, dans cette section, quatre modèles de services (RIMAL 2009), à savoir : (1) Logiciel en tant que services Software as-a-Service (SaaS), où le matériel, l'hébergement, le framework d'application et le logiciel sont dématérialisés, (2) Plateforme en tant que service Platform as-a-Service (PaaS), où le matériel, l'hébergement et le framework d'application sont dématérialisés, (3) Infrastructure en tant que service Infrastructure as-a-Service (IaaS) et (4) Matériel en tant que service Hardware as-a-Service (HaaS), où seul le matériel (serveurs) est dématérialisé dans ces deux derniers cas. La figure 1.4 montre le modèle classique et les différents modèles de service de cloud

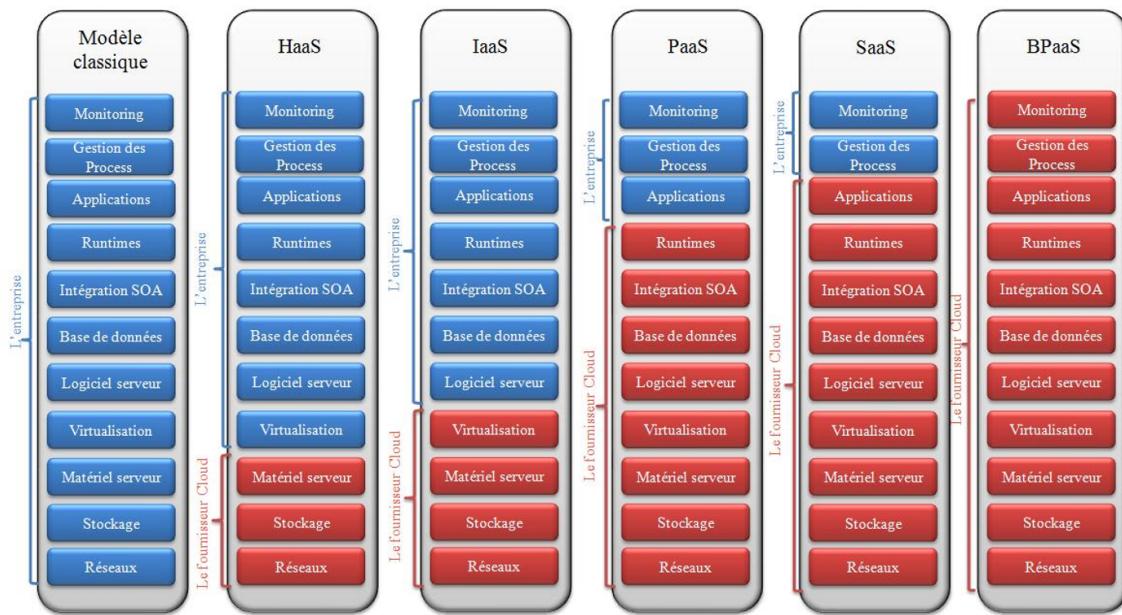


FIGURE 1.4 – Les services XaaS du cloud computing

1.2.4.1 Software as a Service (SaaS)

Ce modèle de service est caractérisé par l'utilisation d'une application partagée qui fonctionne sur une infrastructure Cloud. L'utilisateur accède à l'application par le réseau au travers de divers types de terminaux (souvent via un navigateur web). L'administrateur de l'application ne gère pas et ne contrôle pas l'infrastructure sous-jacente (réseaux, serveurs, applications, stockage). Il ne contrôle pas les fonctions de l'application à l'exception d'un paramétrage de quelques fonctions utilisateurs limitées. On prend comme exemple les logiciels de messagerie au travers d'un navigateur comme Gmail ou Yahoo mail.

1.2.4.2 Platform as a Service (PaaS)

L'utilisateur a la possibilité de créer et de déployer sur une infrastructure Cloud PaaS ses propres applications en utilisant les langages et les outils du fournisseur. L'utilisateur ne gère pas ou ne contrôle pas l'infrastructure Cloud sous-jacente (réseaux, serveurs, stockage) mais l'utilisateur contrôle l'application déployée et sa configuration. Comme exemple de PaaS, on peut citer un des plus anciens - IntuitQuickbase- qui permet de déployer ses applications bases de données en ligne ou -Google App Engine (GAE)- pour déployer des services Web.

Dans ces deux cas l'utilisateur de ces services n'a pas à gérer des serveurs ou des systèmes pour déployer ses applications en ligne et dimensionner des ressources adaptées au trafic.

1.2.4.3 Infrastructure as a Service (IaaS)

L'utilisateur loue des moyens de calcul et de stockage, des capacités réseau et d'autres ressources indispensables (partage de charge, pare-feu, cache). L'utilisateur a la possibilité de déployer n'importe quel type de logiciel incluant les systèmes d'exploitation. L'utilisateur ne gère pas ou ne contrôle pas l'infrastructure Cloud

sous-jacente mais il a le contrôle sur les systèmes d'exploitation, le stockage et les applications. Il peut aussi choisir les caractéristiques principales des équipements réseau comme le partage de charge, les pare-feu, etc. L'exemple emblématique de ce type de service est Amazon Web Services qui fournit du calcul (EC2), du stockage (S3, EBS), des bases de données en ligne (SimpleDB) et quantité d'autres services de base. Il est maintenant imité par de très nombreux fournisseurs.

1.2.5 Modèles de déploiement

Selon la définition du cloud computing donnée par le National Institute of Standards and Technology (NIST) (MELL 2011), il existe quatre modèles de déploiement des services de cloud, à savoir : cloud privé, cloud communautaire, cloud public et cloud hybride, comme illustré dans la figure 1.5.

1.2.5.1 Cloud privé :

L'ensemble des ressources d'un cloud privé est exclusivement mis à disposition d'une entreprise ou organisation unique. Le cloud privé peut être géré par l'entreprise elle-même (cloud privé interne) ou par une tierce partie (cloud privé externe). Les ressources d'un cloud privé se trouvent généralement dans les locaux de l'entreprise ou bien chez un fournisseur de services. Dans ce dernier cas, l'infrastructure est entièrement dédiée à l'entreprise et y est accessible via un réseau sécurisé (de type Virtual Private Network (VPN)). L'utilisation d'un cloud privé permet de garantir, par exemple, que les ressources matérielles allouées ne seront jamais partagées par deux clients différents.

1.2.5.2 Cloud communautaire

L'infrastructure d'un cloud communautaire est partagée par plusieurs organisations indépendantes ayant des intérêts communs. L'infrastructure peut être gérée par les organisations membres ou par un tiers. L'infrastructure peut être située, soit au sein des dites organisations, soit chez un fournisseur de services.

1.2.5.3 Cloud public

L'infrastructure d'un cloud public est accessible à un large public et appartient à un fournisseur de services. Ce dernier facture les utilisateurs selon la consommation et garantit la disponibilité des services via des contrats SLA.

1.2.5.4 Cloud hybride

L'infrastructure d'un cloud hybride est une composition de plusieurs clouds (privé, communautaire ou public). Les différents clouds composant l'infrastructure restent des entités uniques, mais sont reliés par une technologie standard ou propriétaire permettant ainsi la portabilité des données ou des applications déployées sur les différents clouds.

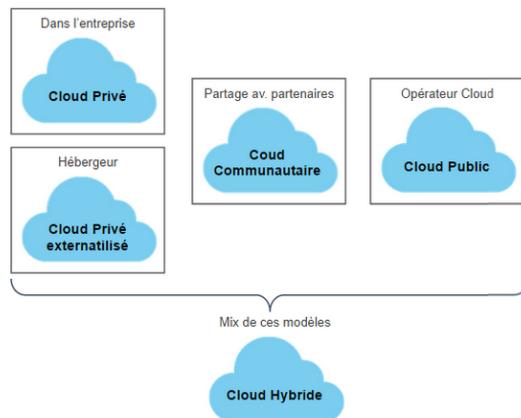


FIGURE 1.5 – Modèles de déploiement du cloud computing

1.2.6 Les principaux grands fournisseurs cloud pour 2019

Comment choisir la solution de cloud la plus adaptée à votre entreprise, quelles sont les différentes offres et les tarifs pour stocker les données informatiques sur un serveur à distance. Comparatif des principales solutions du marché.

1.2.6.1 Amazon Web Services

Avec un chiffre d'affaires de 25.65 milliards de dollar ([Zdnet.fr](#)) réalisé en 2018, le géant de l'IaaS considère 2019 comme une année d'investissement afin d'accélérer le développement de sa technologie et renforcer sa force de vente. En ce début de 2019, Amazon Web Services (AWS) est le leader incontesté dans le domaine de l'IaaS (EPSI 2019).

1.2.6.2 Microsoft Azure

Le service cloud, Microsof Azure a réalisé un chiffre d'affaires annuel de 11 milliards de dollar. En 2019, le service cloud Microsoft Azure vient en deuxième position derrière AWS (IaaS et PaaS). A la différence d'AWS, Microsoft opère plutôt dans le commercial cloud et propose un service très diversifié grâce à une plateforme familiarisée auprès des utilisateurs (EPSI 2019).

1.2.6.3 Google Cloud Platform

L'année 2018 a été particulièrement fructueuse chez le service cloud de Google. Avec un chiffre d'affaires de près 4 milliards de dollar, l'entreprise a décroché en 2018 les contrats les plus importants dans le secteur. Pour 2019, elle va constituer un véritable contrepoint à AWS et à Microsoft même si l'entreprise a toujours du mal à publier convenablement ses rapports financiers jusqu'à présent (EPSI 2019).

1.2.6.4 IBM

Très présent dans le monde du multicloud et du cloud hybride, il dispose actuellement d'un chiffre d'affaires de 12.2 milliards de dollar. Même s'il reste un peu derrière AWS et Microsoft, il est en train de se focaliser sur le développement d'une

plateforme capable de gérer plusieurs outils d'intelligence artificielle pour les principaux fournisseurs cloud. En plus, l'acquisition de Red Hat est un signal fort pour le cloud hybride (EPSI 2019).

1.3 Workflow

1.3.1 Introduction au Workflow

Un Workflow est la modélisation et la gestion assistée par ordinateur de l'accomplissement des tâches composant un processus administratif ou industriel, en interaction avec divers acteurs (humains, logiciels, ou matériels) invoqués (COURTOIS 1996). Outil informatique d'origine industrielle, le Workflow est l'adaptation de la GED24 adjoint de la faculté à gérer l'échange de messages. Le Workflow propose des solutions d'optimisation et de rationalisation des flux d'informations ; que ces informations soient associées à des documents, des procédures ou des messages complémentant les systèmes de gestion électronique de documents et d'informations.

A l'heure actuelle plus de 250 Systèmes de Gestion de Workflow (WFMS) sont utilisés ou en développement. Cela signifie que le terme « gestion de Workflow » n'est pas simplement une nouvelle expression à la mode. Ce phénomène de gestion de processus (Workflow) aura certainement un fort impact sur la génération suivante de systèmes informatiques (COURTOIS 1996).

1.3.2 Origines

Il est intéressant de considérer l'évolution des systèmes informatiques au cours des quatre dernières décennies (VANDERAALST 1997) pour prendre conscience de la pertinence d'une gestion électronique de processus (Workflow) et apprécier l'impact de la gestion de Workflow dans un avenir proche.

La Figure 1.6 présente le phénomène de gestion de Workflow dans une perspective historique. Cette figure décrit l'architecture d'un système informatique classique en termes de composants. Dans les années soixante, un système informatique était composé d'un certain nombre d'applications autonomes. Pour chacune de ces applications une interface utilisateur et un système de base de données spécifique étaient développés, chaque application possédait donc ses propres routines pour interagir avec l'utilisateur, stocker et récupérer les données. Dans les années soixante-dix, le développement des Système de Gestion de Bases de Données (SGBD) a permis d'extraire les données des applications. En utilisant les SGBD, les applications ont ainsi été libérées du fardeau de la gestion de données. Dans les années quatre vingts, l'apparition de systèmes de gestion d'interface utilisateur User Interface Management Systems (UIMS) a permis aux développeurs d'application d'extraire l'interaction avec les utilisateurs des applications.

Enfin, les années quatre-vingt-dix sont marquées par l'apparition de logiciels de Workflow, permettant aux développeurs d'application d'extraire les procédures de travail des applications. La Figure 1.6 fait apparaître le système de gestion de Workflow comme une composante générique pour représenter et manipuler les processus d'entreprise¹.

1. Les procédures d'entreprise représentent l'organisation et la politique de l'entreprise pour atteindre certains objectifs (WFMC 1999)

Ainsi, à l'heure actuelle, beaucoup d'organisations commencent à considérer l'utilité d'outils avancés pour soutenir la conception et l'exécution de leurs processus d'entreprise.

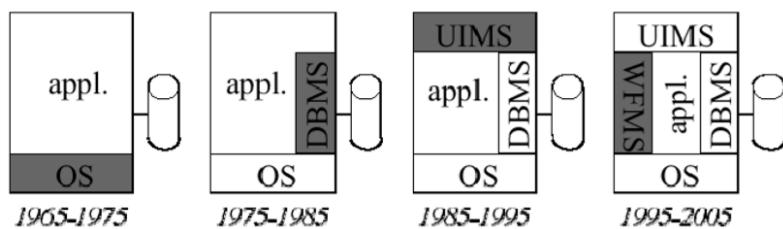


FIGURE 1.6 – Les systèmes de gestion de Workflow dans une perspective historique

1.3.3 Définitions et terminologies

Les définitions sont, pour la majorité, issues de la Coalition de Gestion de Workflow "Workflow Management Coalition (WfMC)". La WfMC a été fondée en 1993 par un regroupement d'industriels de l'informatique, de chercheurs et d'utilisateurs, associée à l'essor du développement des Workflows. Cette coalition a pour but de promouvoir les Workflow et d'établir des standards pour les WorkFlow Management System (WFMS). Elle a en particulier publié un glossaire de référence contenant les terminologies employées dans ce domaine (WfMC 1999). Ces standards servent notamment à résoudre les problèmes d'interopérabilité entre systèmes Workflow mais également à définir les caractéristiques fondamentales de ces systèmes. Les documents publiés par la WfMC, qui couvrent plusieurs aspects, peuvent être considérés comme des références en la matière.

1.3.4 Définitions de base du Workflow

Le sens du mot Workflow peut varier en fonction du contexte. Pour plus de clarté, les définitions les plus communément admises sur les concepts et les termes du Workflow sont rappelées ci dessous. Ces définitions sont principalement issues du Workflow Management Coalition Terminology and Glossary (WFMC-TC) (WFMC 1999), dont il existe une traduction à usage francophone (WFMC 1999). L'idée première du Workflow est donc de séparer les processus, les ressources et les applications, afin de se recentrer sur la logistique des processus travail et non pas sur le contenu des tâches individuelles. Un Workflow est donc le lien entre ces trois domaines comme précise la Figure 1.7.

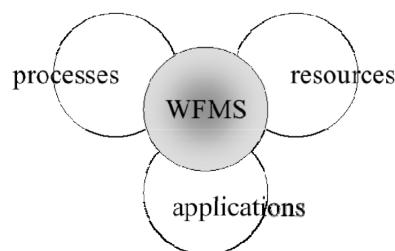


FIGURE 1.7 – Environnement du système Workflow (ZACHAREWICZ 2006)

1.3.4.1 Définition d'un Workflow

Le Workflow est une technologie informatique ayant pour objectif la gestion des processus d'organisations ou d'entreprises : les termes suivants sont également employés pour qualifier cette technologie « Système de Gestion Electronique de Processus », « Gestion de Workflow » ou « Gestion de processus » (COURTOIS 1996).

Le Workflow est l'ensemble des moyens mis en œuvre pour automatiser et gérer les processus d'une organisation. Cette gestion est rendue possible par la représentation sous forme d'un modèle, de tout ou partie des processus considérés. Le Workflow doit ensuite transcrire les modèles obtenus en une forme exécutable. Enfin, ces modèles sont exécutés et gérés. Il est ainsi possible de suivre l'évolution de leur état au fil du temps. La gestion de processus inclut également, au cours de l'exécution, la coordination et la synchronisation des différents acteurs des processus en fonction de l'état actuel des modèles.

Pour résumer, la Gestion de Processus permet donc d'attribuer à chacun et au bon moment, les tâches dont il a la responsabilité et de mettre à disposition les applications, les outils et les informations nécessaires pour leurs réalisations. Dans un contexte d'acteurs humains, le Workflow permet de décharger les acteurs de certaines tâches de gestion administrative, en leur laissant la possibilité de se concentrer sur les contenus des tâches techniques en rapport avec leurs compétences. De plus, le Workflow donne la possibilité d'effectuer une activité de monitoring sur le déroulement des Workflow de l'entreprise, permettant en particulier de connaître, en fonction de la date, l'état des activités, des acteurs, des applications et quelles sont les prochaines activités planifiées.

En synthèse, La WfMC présente le Workflow comme l'automatisation d'un processus d'entreprise, en intégralité ou en partie, pendant laquelle on définit les transmissions des documents, de l'information ou des tâches d'un participant à un autre pour agir, selon un jeu de règles procédurales (WFMC 1999). Un Système Workflow définit, gère et exécute des procédures en exécutant des programmes dont l'ordre d'exécution est prédéfini dans une représentation informatique de la logique de ces procédures - les Workflow (WFMC 1999).

1.3.4.2 Dimensions

Associé aux définitions précédentes, il a été défini une représentation tridimensionnelle des systèmes Workflow (VANDERAALST 1997) (Figure 1.8).

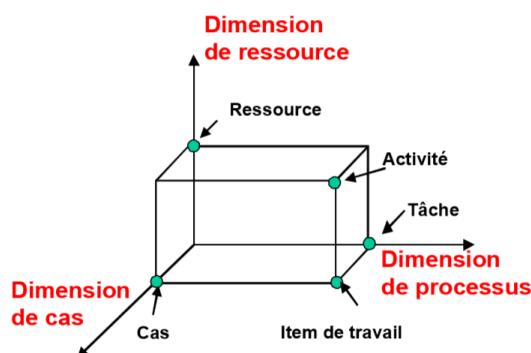


FIGURE 1.8 – Dimensions des systèmes Workflow

La Figure 1.8 donne une représentation tridimensionnelle d'un Workflow : avec une dimension de cas, une dimension de processus et une dimension de ressource. La dimension de cas représente l'instanciation du Workflow, chaque cas est un modèle à simuler et traiter individuellement. Les cas ne s'influencent donc pas directement. Ils peuvent éventuellement s'influencer indirectement via les ressources et les données. Le processus de Workflow est spécifié dans la dimension de processus, c'est-à-dire, la définition des tâches et de leur enchaînement. Enfin, dans la dimension de ressource, les ressources sont regroupées en rôles et en unités organisationnelles. En résumé, nous pouvons donc visualiser un certain nombre de termes Workflow dans la vue tridimensionnelle de la Figure 1.9 . En particulier, une entité de travail est définie par la coordonnée cas + tâche et une activité par le cas + la tâche + la ressource. Cette représentation met en relief la gestion de Workflow comme le lien entre les cas, les tâches et l'organisation.

1.3.5 Description des Systèmes Workflow

Après avoir présenté la terminologie Workflow, il est maintenant possible de présenter le principe de fonctionnement des systèmes Workflow. A un haut niveau, ce type de système est composé de trois parties fonctionnelles (WFMC 1999). La Figure 1.9 illustre un Système de gestion de Workflow et les relations entre ses différentes fonctions.

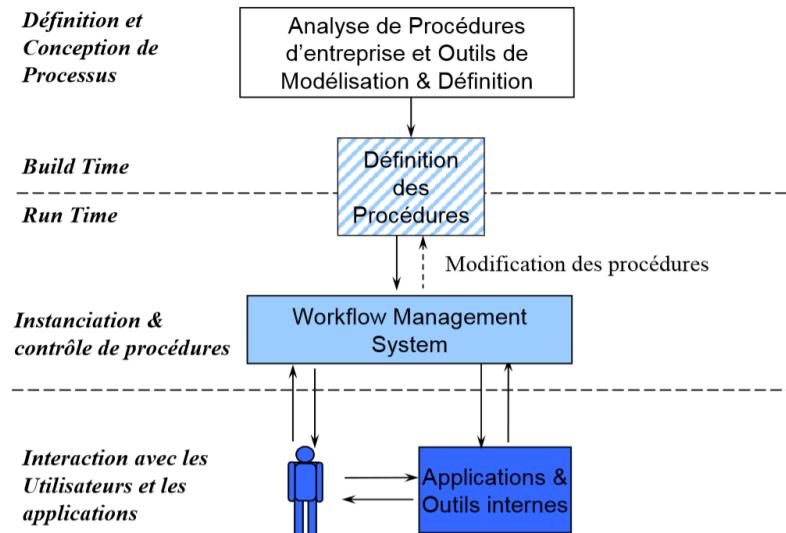


FIGURE 1.9 – Caractéristiques du système Workflow

1.3.5.1 Build time

Cette première phase permet la définition et la modélisation des procédures Workflow, elle est nommée build time. Elle est principalement composée d'un outil permettant la modélisation des Workflow dans un formalisme existant ou propriétaire à partir d'une analyse de procédures d'entreprises, il est à noter que les outils actuels préfèrent une description graphique. Un deuxième composant transcrit les modèles obtenus dans une définition des procédures « exécutable », c'est à dire compréhensibles par la partie chargée de l'exécution (simulateur du run time). Certains

systèmes permettent en retour de la partie run time des modifications dynamiques de la structure de définition de procédures comme indiqué sur la Figure 1.9.

1.3.5.2 Run time

L'environnement d'exécution (ou run time) a l'entièrre gestion des modèles de Workflow établis dans la première partie. Cette gestion comprend l'exécution des Workflow avec un simulateur mais aussi la distribution des tâches aux rôles appropriés en cours d'exécution, la mise à disposition de l'ensemble des données et outils nécessaires, la supervision et le contrôle, etc.

Plus en détails, cette partie se décline en sous composants, introduits ci-dessous : Le Moteur de Workflow (Workflow Engine) qui est chargé de la Gestion des Procédures à travers la simulation de leurs évolutions.

Le gestionnaire des listes de tâches (Worklist Handler), chargé de distribuer les activités dans les listes des acteurs en fonction de leurs rôles.

Les listes de tâches (Worklists), qui sont les listes associées à chaque rôle dans lesquelles le moteur place les tâches à réaliser. Les tâches sont classées par priorité ou par date avec les données et les outils à utiliser de façon appropriée.

Les outils d'administration et de contrôle (Workflow Process Monitoring) suivent le déroulement des procédures Workflow, elles peuvent fournir l'état actuel des composantes du Workflow et donner l'ordre de modifier le modèle de procédures.

1.3.5.3 Utilisateurs, outils et applications

La troisième phase concerne l'ensemble des outils et des fonctions d'API (Application Programming Interface) qui permettent au système Workflow de s'interfacer avec des ressources humaines, des applications informatiques extérieures et d'autres systèmes Workflow.

1.4 Classification des systèmes Workflow

Il n'existe pas de classification commune des systèmes Workflow dans la littérature, reconnue par l'ensemble de la communauté Workflow (VANDERAALST 1997). Ceci étant essentiellement dû au nombre important de critères de classification qu'il est possible de retenir. En effet, les spécialistes adoptent différents points de vue par rapport à la notion de Workflow, les critères qui en découlent varient donc en fonction de leurs perceptions des caractéristiques présentées par ces systèmes Workflow. Ainsi, il existe plusieurs classifications, permettant de sélectionner un outil de gestion de Workflow avec différents « éclairages » sur le sujet.

Malgré ce manque d'unité, la classification proposée par (MCREADY 1992) est assez répandue dans la littérature, elle est reprise par bon nombre d'auteurs (VANDERAALST 1997), (GEORGAKOPOULOS 1995). Elle propose de distinguer quatre catégories de Systèmes Workflow. La Figure 1.10 présente ces différentes classes selon deux axes : Approche et Structure.

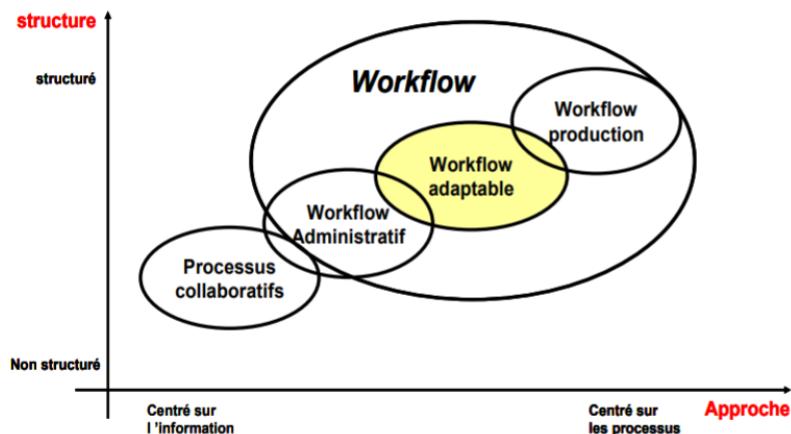


FIGURE 1.10 – Différentes classes des systèmes Workflow

1.4.1 Processus collaboratifs

Cette première classe est axée sur la communication et sur le partage d'information. Les systèmes collaboratifs sont définis pour supporter le travail en groupe, dans le cadre de la conception, de la gestion de projet ou de la résolution de problèmes faisant appel à plusieurs niveaux d'expertise. Ces systèmes permettent de réunir les intervenants d'un projet autour d'un objectif commun, les clients de la procédure y étant souvent eux-mêmes directement associés, les logiciels employés sont le plus souvent des groupwares¹. Les tâches des procédures gérées sont le plus souvent complexes et leur réalisation implique l'intervention de ressources aux compétences très spécifiques pour une forte valeur ajoutée. D'un autre côté, l'enchaînement des activités des procédures à traiter est faiblement structuré et peu répétitif. De part la faible structure de ces processus, ils ne font pas partie ou se situent à la frontière de ce que l'on considère comme la « sphère » Workflow comme le précise la Figure 1.10.

1.4.2 Workflow administratif

Les systèmes Workflow administratifs (General Purpose Workflow Management Systems) ont pour objectif de décharger les ressources d'une entreprise des tâches administratives. En effet ces procédures sont répétitives, fortement prédictibles et les règles d'enchaînement des tâches sont très simples et clairement définis ; ces procédures sont donc aisément automatisables, évitant ainsi un travail fastidieux où peuvent naître des erreurs souvent humaines. Les systèmes Workflow administratifs permettent de lier à une tâche administrative, les documents et les informations nécessaires à la réalisation de cette tâche par un acteur humain. Ces systèmes gèrent également le routage des documents et le remplissage de formulaires. La gestion par Workflow de procédures administratives permet un gain de l'ordre de 5% à 10% en termes de productivité et de 30 à 90% en termes de délais (ADER 1996). Enfin une dernière raison de l'automatisation de ce type de procédures en Workflow provient du fait que ces procédures possèdent une structure statique et ne sont donc pas

1. Collecticiel : classe de logiciels prévus pour être exploités de façon concurrente, sur un même projet.

souvent assujetties à modifications car elles possèdent une longue durée d'utilisation (SCHEER et TRAUT. 1997).

1.4.3 Workflow de production

Les systèmes Workflow de production impliquent des procédures prévisibles et assez répétitives. Leurs principales différences avec les Workflow administratifs résident dans la complexité des tâches et de la structure des procédures, dans leur capacité à faire appel à des informations provenant de systèmes d'information variés et dans l'enjeu que représente leur réussite. En effet, la procédure Workflow correspond directement au travail effectué par l'entreprise. En d'autres termes, la performance de l'entreprise est directement liée à l'exécution de la procédure managée par le Workflow. On dit dans ce cas qu'il est mission critical¹ (INCONCERT 1997).

1.4.4 Workflow adaptable ou Workflow ad hoc

L'impossibilité pour les systèmes de gestion de Workflow traditionnels de traiter les différents changements dynamiques dans les flux de travail est une limite à dépasser. A ce titre, il a été introduit les concepts de Workflow adaptable (adaptive Workflow) (VANDERAALST 1997) et de Workflow ad hoc (VOORHOEVE 1997). La nuance entre ces deux nouveaux termes provient du fait qu'ad hoc désigne un acte spécialement fait pour un objet déterminé alors qu'adaptable prévoit un changement définitif de la procédure.

Les Workflow ad hoc se situent à la frontière gauche de la représentation Figure 1.10 dans la « sphère » Workflow adaptable. Ils régissent des procédures dont la structure est déterminée pendant l'exécution en fonction des décisions humaines prises suite à la réalisation d'une tâche, plus concrètement, la structure se construit par pas en suivant le rythme de l'exécution. En effet, la réalisation d'une procédure non structurée peut impliquer à chaque fois l'exécution d'un nouvel enchaînement des tâches, voire la création de nouvelles tâches. Il n'y a pas a priori de persistance de l'enchaînement de ces tâches.

Les Workflow adaptables sont, quant à eux, des supports comparables aux Workflow de production classiques possédant une structure préétablie, mais pouvant traiter certains changements de structure « en ligne ». Ces changements peuvent aller des changements individuels/ad hoc (gestion d'exception), c'est à dire d'un aiguillage pour déterminer l'activité suivante, jusqu'à la reconception par Business Process Reengineering (BPR) de processus (VANDERAALST 1997). En conclusion, ils ont une action globale pouvant inclure la définition du Workflow ad hoc

Il est intéressant de classifier les différents changements possibles par un Workflow adaptables, dans le but de mieux les anticiper (SADIQ 1999). Les changements sont envisageables selon plusieurs perspectives : la ressource, le contrôle, la procédure, la tâche et le système (VANDERAALST 1997).

1. Un système est dit critique lorsque : les vies de personnes sont tributaires de son fonctionnement ou le coût économique d'un dysfonctionnement est catastrophique.

1.4.5 Modèle de référence des systèmes Workflow

Le modèle de référence, Figure 1.11, présente l'architecture générale de l'environnement proposée par la WfMC, il identifie les interfaces couvrant cinq domaines de fonctionnalités entre le système Workflow et son environnement.

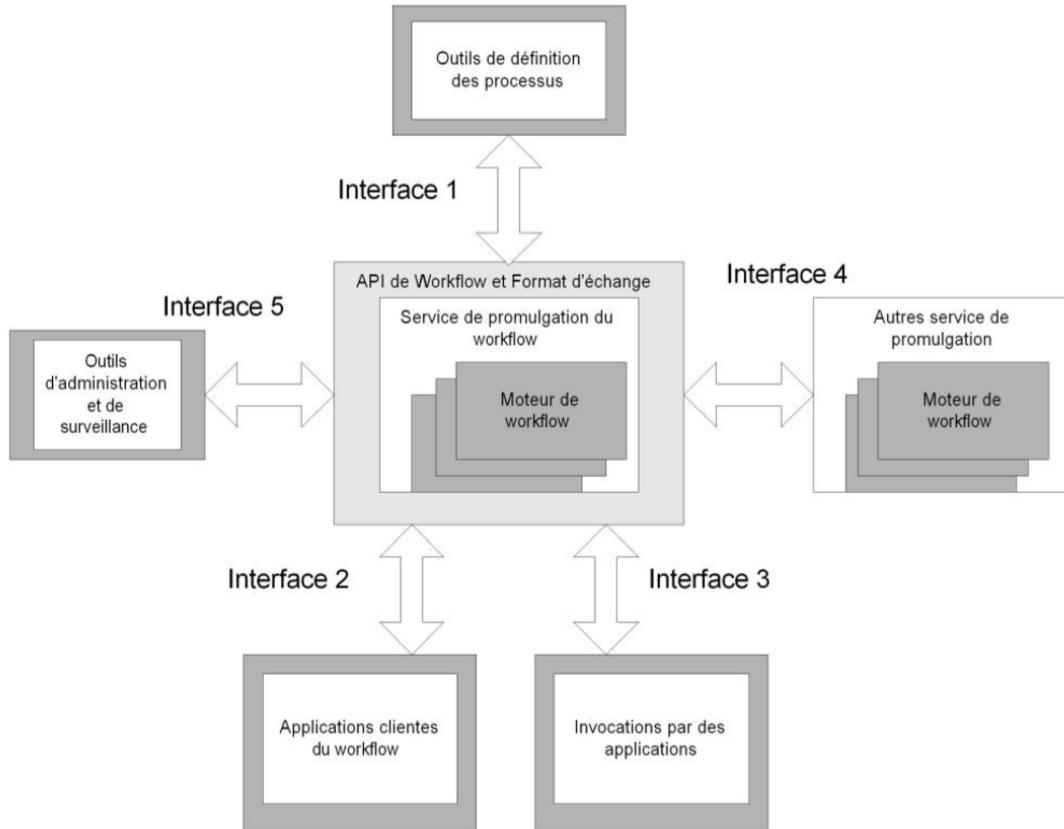


FIGURE 1.11 – Modèle de référence des systèmes de gestion de workflow (WfMC, 95).

- **Interface 1 :** Correspond à l'échange des modèles entre les moteurs de workflows et les différents outils de modélisation de processus.
- **Interface 2 :** Permet à des applications clientes de communiquer avec le moteur de workflows.
- **Interface 3 :** Permet au système de workflow d'invoquer des applications clientes.
- **Interface 4 :** Permet l'interopérabilité entre les différents moteurs de workflows.
- **Interface 5 :** Correspond à l'interaction entre les applications d'administration et de pilotage et le moteur de workflows.

1.5 Intérêt du cloud pour les workflows

Les clouds offrent plusieurs avantages pour les applications à base de workflows. Ces avantages facilitent :

1.5.1 L'approvisionnement de ressources

Dans les grilles, l'ordonnancement est basé sur un modèle en best-effort, dans lequel l'utilisateur spécifie la quantité de temps nécessaire et délègue la responsabilité de l'allocation des ressources et d'ordonnancement de tâches à un ordonnanceur fonctionnant en mode batch utilisant des files d'attentes. Dans le cloud, au lieu de déléguer l'allocation au gestionnaire de ressources, l'utilisateur peut provisionner les ressources nécessaires et ordonner les tâches en utilisant un ordonnanceur contrôlé par l'utilisateur. Ce modèle d'approvisionnement est idéal pour les workflows, car il permet au système de gestion de workflow d'allouer une ressource une seule fois et de l'utiliser pour exécuter de nombreuses tâches.

1.5.2 L'allocation dynamique de ressources à la demande

Contrairement aux grilles, les clouds donnent l'illusion que les ressources informatiques disponibles sont illimitées. Cela signifie que les utilisateurs peuvent demander, et s'attendre à obtenir des ressources suffisantes pour leurs besoins, à tout moment. L'approvisionnement à la demande est idéal pour les workflows et d'autres applications faiblement couplées, car il réduit le surcoût (overheads) d'ordonnancement total et peut améliorer considérablement les performances du workflow (Singh, 2005 ; Juve, 2008)

1.5.3 L'élasticité

Outre l'approvisionnement des ressources à la demande, les clouds permettent aussi aux utilisateurs de libérer des ressources à la demande. La nature élastique de clouds facilite le changement des quantités et des caractéristiques de ressources lors de l'exécution, permettant ainsi d'augmenter le nombre de ressources, quand il y a un grand besoin, et d'en diminuer, lorsque la demande est faible. Cela permet aux systèmes de gestion de workflow de répondre facilement aux exigences de qualité de service (QoS) des applications, contrairement à l'approche traditionnelle, qui nécessite de réservier à l'avance des ressources dans les environnements de grilles.

1.5.4 La garantie des QoS via des SLA

Avec l'arrivée des services de cloud computing provenant de grandes organisations commerciales, les accords de niveau de service (SLA) ont été une préoccupation importante pour les fournisseurs et les utilisateurs. En raison de compétitions entre les fournisseurs de services émergents, un grand soin est pris lors de la conception du SLA qui vise à offrir (i) de meilleures garanties de QoS aux utilisateurs, et (ii) des termes clairs pour l'indemnisation, en cas de violation du contrat. Cela permet aux systèmes de gestion de workflow de fournir de meilleures garanties de bout en bout en "mappant" les utilisateurs aux fournisseurs de services selon les caractéristiques des SLA.

1.5.5 Le faible Coût d'exploitation

Économiquement motivés, les fournisseurs de cloud commercial s'efforcent d'offrir de meilleures garanties de services par rapport aux fournisseurs de grille. Les

fournisseurs de cloud profitent également des économies d'échelle, en fournissant des ressources de calcul, de stockage et de bande passante, à un coût très faible grâce, à la virtualisation. Ainsi l'utilisation des services de cloud public pourrait être économique et une alternative moins coûteuse, par rapport à l'utilisation de ressources dédiées, qui sont plus chères. Un des avantages de l'utilisation des ressources virtuelles pour l'exécution de workflow, plutôt que d'un accès direct à la machine physique, est le besoin réduit pour sécuriser les ressources physiques des codes malveillants. Cependant, l'effet à long terme de l'utilisation de ressources virtuelles dans les clouds qui partagent efficacement une "tranche" de la machine physique, plutôt que d'utiliser des ressources dédiées pour les workflows de calculs intensifs, est une question de recherche intéressante.

1.6 Modélisation de workflow avec les réseaux de pétri

Les Réseaux de Petri (RdPs) constituent un formalisme majeur pour modéliser les processus workflows. Une des forces des RdPs est la base mathématique forte qu'ils offrent avec une représentation graphique.

1.7 les réseaux de Pétri

1.7.1 Qu'est-ce que les réseaux de Pétri

Les réseaux de Pétri sont définis comme étant un formalisme qui permet la description et l'analyse du comportement des systèmes concurrents, introduit par Carl Adam Pétri en 1962. (RDP 2008)

Définition 1.7.1. (Définition d'un réseau de pétri :)

Un réseau de pétri (R) est un triple $R = (P, T, W)$ où P est l'ensemble des places (les places représentent les sites) et T l'ensemble des transitions (les transitions représentent les actions) tel que :

- P est un ensemble final de places,
- T est un ensemble final de transition ($P \cap T = \emptyset$),
- $W : (P \times T) \cup (T \times P) \rightarrow N = \{0, 1, 2, \dots\}$.

1.7.2 Représentation d'un réseau de Pétri

1.7.2.1 Représentation graphique

L'un des aspects les plus agréables des réseaux de Pétri est qu'il est extrêmement aisés de les visualiser ; c.-à-d., donner une interprétation graphique à sa structure qui peut être représentée à travers un graphe bipartie fait de deux types de sommets : les places et les transitions reliées alternativement par des arcs orientés qui portent des poids entiers positifs, si un poids n'est pas porté alors il est égal à 1 (RdP ordinaire). Généralement, les places sont représentées par des cercles et les transitions par des rectangles, le marquage d'un RdP est représenté par la distribution de jetons dans l'ensemble de ses places telle que chaque place peut contenir un ou plusieurs jetons représentés par des points dans le cercle représentant la place. (BELOUNNAR 2011)

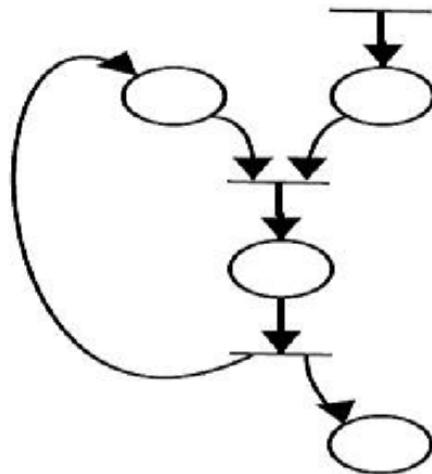


FIGURE 1.12 – Réseau de Pétri simple

1.7.2.2 Représentation matricielle

Une représentation matricielle d'un RdP est offerte afin de simplifier les Tâches d'analyse et de vérification effectuée sur un modèle RdP. Agir sur une représentation graphique d'un modèle RdP est une Tâche délicate en comparant avec une représentation matricielle. Il est possible de représenter la fonction W (fonction de poids) par des matrices.(RDP 2008)

Définition 1.7.2. (Réseau de pétri) : Soit Un réseau de pétri $R = (P, T, W)$ avec $P = \{p_1, p_2, \dots, p_m\}$ et $T = \{t_1, t_2, \dots, t_n\}$, on appelle matrice des pré conditions pré la matrice $m \times n$ à coefficients dans N tel que $pre(i, j) = W(p_i, t_j)$, elle indique le nombre de marques que doit contenir la place p_i pour que la transition t_j devienne franchissable, de la même manière on définit la matrice des post conditions post la matrice $n \times m$ tel que $post(i, j) = W(t_j, p_i)$ contient le nombre de marques déposées dans p_i lors du franchissement de la transition t_j . La matrice $C = \text{post} - \text{pré}$ est appelée matrice d'incidence du réseau (m représente le nombre de places d'un réseau de Pétri et n le nombre de transitions.).

La représentation Graphique d'un marquage dans un RdP marqué est présentée par des marques dans la place appelées jetons.

Le marquage d'un réseau de Petri est représenté par un vecteur de dimension m à coefficients dans N. La règle de franchissement d'un réseau de Petri est définie par : $M'(p) = M(p) + C(p, t)$.

1.7.2.3 Représentation d'un RdP marqué

Un réseau de Pétri marqué est le couple $N = < R, M >$ où :

- R est un réseau de Pétri.
- M est une application de marquage.
- $M : P \rightarrow N$.

$M(p)$ est le nombre de marques (jetons) contenus dans la place p. Le marquage d'un réseau de Pétri est une opération qui consiste à assigner des jetons dans les places.

On appelle marquage M d'un Réseau de Pétri le vecteur du nombre de marques dans chaque place : la I^{eme} composante correspond au nombre de marques dans la I^{eme} place. Il indique à un instant donné l'état du RdP.(RDP 2008)

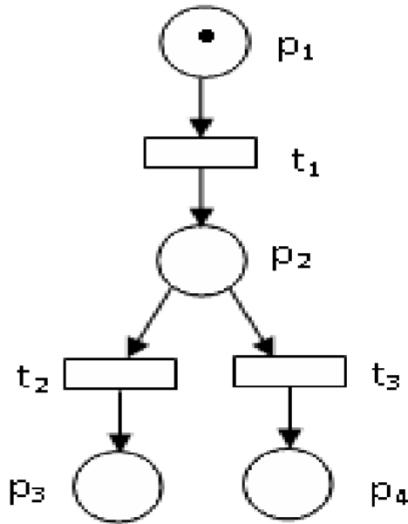


FIGURE 1.13 – Réseau de Pétri marqué

Example 1.7.1. Pour le réseau de la (Figure 1.13).

$$P = \{p_1, p_2, p_3, p_4\} \text{ et } T = \{t_1, t_2, t_3\}$$

$$pre = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$post = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(a) La matrice pré

$$C = \begin{pmatrix} -1 & 0 & 0 \\ 1 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(c) La matrice d'incidence

(b) La matrice post

$$M = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

(d) Le vecteur de marquage M

Le marquage du RdP présenté à la Figure 3 est donné par :

$$M = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

On appelle marquage initial, noté M_0 , le marquage à l' instant initial ($t = 0$).

1.7.3 réseaux workflow (WF-nets)

Définition 1.7.3. (WF-nets) : Le réseau de pétri qui permet de modéliser le contrôle de flux d'un workflow est appelé un WF-net (réseaux workflow) Ssi :(SBAÏ 2010)

1. Il existe une seule place source $i \in P$ avec $*i = \emptyset$. (i début du processus).
2. Il existe une seule place puits $o \in P$ avec $*o = \emptyset$ (o est la fin du processus).
3. Chaque nœud $x \in (P \cup T)$ est sur un chemin de i à o . C'est à dire : Chemin $(i, x) \wedge$ Chemin (x, o) .



FIGURE 1.15 – Procédure modélisée avec le WF-net

Un réseau de workflow (WF-net) est un réseau de pétri qui possède une seule place d'entrée (i) et une seule place de sortie (o) et puis que n'importe quel cas traité avec la procédure représentée par un WF-net est créée au moment de début de son traitement par le système de gestion de workflow (SGWF) est supprimé une fois que sont traitement est complètement achevé par le SGWF. Cela veut dire que le WFnet spécifie le cycle de vie d'un cas du processus modélisé. (SBAÏ 2010)

Remarque 1.7.1. *Un chemin C dans un WF-net d'un nœud n_1 au nœud n_k est une séquence $< n_1, n_2, \dots, n_k >$ tel-que $< n_i, n_{i+1} > \in F$, pour $i = 1 \dots k - 1$. F est la relation de flux.*

1.7.4 Constructions du routage dans un WF-net

Un réseau de workflow (WF-net) est utilisé pour spécifier le routage de flux dans les cas du workflow, quatre types de routage ont été identifiés :(SBAÏ 2010)

1.7.4.1 Séquentiel :

Utilisé pour traiter la relation de causalité entre les tâches. Soit deux tâches A et B, si la tâche B est exécutée après l'accomplissement de A, ce comportement peut être modélisé avec un RDP en ajoutant les places pour capter les relations de causalité entre les tâches A et B. La figure 1.16 illustre ce comportement.

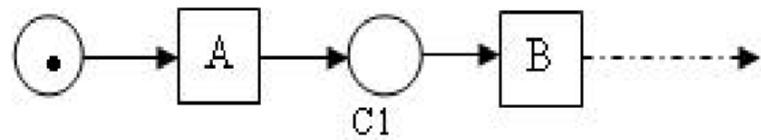


FIGURE 1.16 – Routage séquentiel.

1.7.4.2 Parallèle :

Utiliser dans le cas où l'ordre d'exécution est moins strict. Par exemple si on a deux tâches B et C qui doivent être exécutées mais dans un ordre arbitraire, la modélisation de ce comportement de routage parallèle comporte deux constructions le AND-split et AND-join.

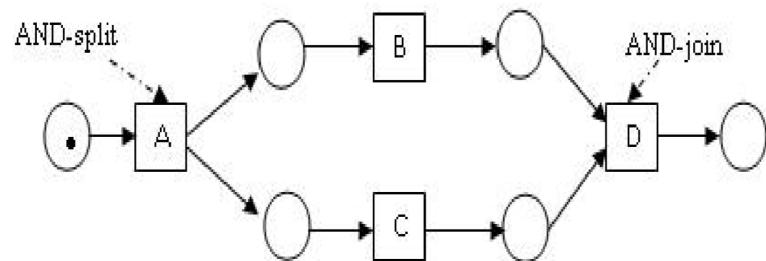


FIGURE 1.17 – Routage parallèle.

1.7.4.3 Conditionnel :

Utilisé pour traiter les cas où le routage de flux peut dépendre des données d'un cas. Il existe deux types de routage conditionnel : *le choix non-déterministe* et *le choix déterministe*.

1. **Choix non-déterministe** : Pour modéliser ce comportement deux constructions sont utilisées OR-Split et OR-join. La figure 1.18 représente un choix entre l'exécution des deux tâches B ou C grâce à un jeton dans la place c2, mais cela permet un choix non-déterministe (l'exécution de B ou de C car les deux tâches sont permises).

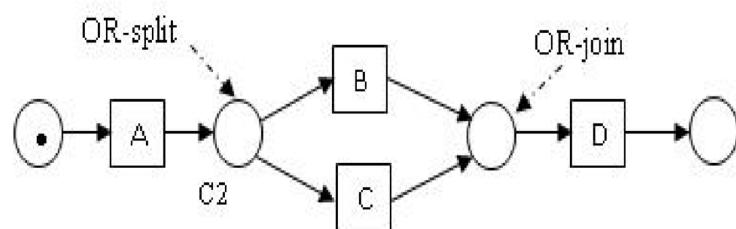


FIGURE 1.18 – Routage conditionnel non-déterministe

2. Choix déterministe : Le choix entre plusieurs alternatives peut être guidé par les données (ou les attributs) du workflow, dans ce cas c'est un choix déterministe.

Nous pouvons prendre le schéma de la figure précédente et ajouter une pré-condition pour chacune des tâches B et C (en se basant sur un ou plusieurs attributs du workflow).

Il y a une autre façon de modéliser le choix déterministe comme dans la figure suivante. La transition A comporte deux places de sortie c2 et c3 et produit un seul jeton (soit dans c2 ou dans c3). Le choix entre les deux places est basé sur la valeur de l'attribut x. Un symbole spécial est utilisé pour représenté que la tâche A est un OR-split (exclusive OR).

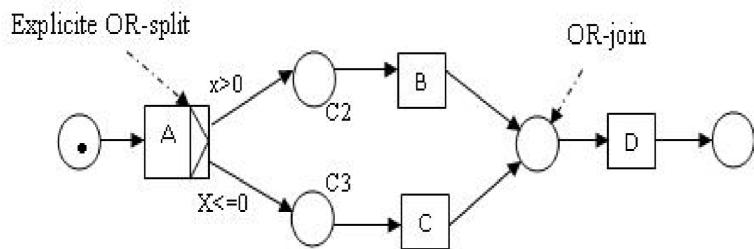


FIGURE 1.19 – Choix explicite entre B et C à base de l'attribut x.

1.7.4.4 Itération :

L'itération peut être modélisée en utilisant l'implicite (OR-split, OR-join), l'explicite (OR-split, OR-join). Il est utilisé dans le cas où une ou plusieurs tâches peuvent être exécutées plusieurs fois. La figure suivante montre un exemple d'un modèle de workflow en utilisant un routage itératif.

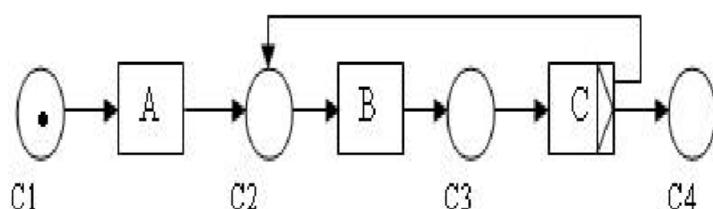


FIGURE 1.20 – Itération : B peut être exécutée plusieurs fois

1.8 Conclusion

Dans ce chapitre, nous avons étudié la théorie sur les concepts fondamentaux de cloud et de workflow.

Dans le chapitre suivant, nous expliquerons l'étape d'analyse et la conception du système de flux de travail administratif, en prenant en compte l'étude théorique.

Chapitre 2

Analyse et Conception

Chapitre 2

Analyse et Conception

Introduction

Dans ce chapitre nous aborderons une description générale de notre application, ensuite nous mettons en évidence le côté conceptuel de nos applications qui constitue une étape fondamentale qui précède l'implémentation, cette étape permet de détailler les différents diagrammes et scénarios à implémenter dans la phase suivante. Ceci permettra de mieux comprendre nos applications. Nous avons utilisé une démarche générale basée sur Unified Process (UP) qui utilise Unified Modeling Language (UML).

Dans cette démarche, les diagrammes UML utilisés sont les suivants : Diagramme de cas d'utilisation, le diagramme de classes et le diagramme d'activité sont modélisés à l'aide de l'outil ***Lucidchart***.

2.1 Unified Modeling Language (UML) :

Dans ce qui suit, nous allons donner une brève description d'UML.

2.1.1 Définition UML

L'OMG définit l'UML comme un langage visuel dédié à la spécification, la construction et la documentation des artefacts d'un système logiciel. Aussi la façon dont tout le monde modélise non seulement la structure de l'application, le comportement et l'architecture, mais aussi des processus d'affaires et la structure des données. Ce langage est conçu pour modéliser divers types de systèmes et de taille quelconque. Il possède une approche entièrement objet couvrant tout le cycle de développement. Le système est décomposé en un ensemble d'objets collaborant (MG 2019).

2.1.2 Contenu UML

L'UML comporte 13 diagrammes qui se répartissent en deux catégories (MG 2019). Nous ne mentionnons que les diagrammes que nous utilisons,

2.1.2.1 Diagramme structurel :

- **Diagramme de classes (Class Diagram)** : ce diagramme décrit la structure statique du système, il définit les classes, leurs attributs et leurs relations. Il est considéré comme le diagramme le plus important.
- **Diagramme de paquetages (Package Diagram)** : définit les dépendances entre les paquets (groupement d'éléments UML) constituant un modèle.

2.1.2.2 Diagramme comportemental :

- **Diagramme de cas d'utilisation (Use Case Diagram)** : pour décrire les besoins des utilisateurs.
- **Diagramme de séquence (Sequence Diagram)** : décrit comment chaque objet interagit avec l'autre et dans quel ordre, sur un axe temporel donné. Ce diagramme est associé aux diagrammes de cas d'utilisation..

2.2 Identification des objectifs

2.2.1 Spécifications fonctionnelles

Le système doit permettre à l'administrateur de :

1. Générer le modèle.
2. Gérer les sous-directions.
3. Gérer les services.
4. Gérer workflow pour les dossiers.
5. Gérer les tâches des services.
6. Gérer les utilisateurs.
7. Consulte les historiques et la recherche par tâche.
8. Consulte les historiques et la recherche par dossier et les jours pour chaque tâche.
9. Consulte bordereau.

Le système doit permettre à l'utilisateur de :

1. Gérer les dossiers.
2. Activé le dossier.
3. Finir le traitement des dossiers par tâche.
4. Gérer les bordereaux.
5. Accepté le bordereau.
6. Refuse le bordereau.
7. Imprimer le bordereau.

2.2.2 Spécifications techniques

1. Le système doit être déployé sur le cloud.
2. La configuration et la génération de code dynamique.
3. Modélisation d'un workflow Administratif via une interface graphique simple et utile.

2.2.3 Diagramme de cas d'utilisation

2.2.3.1 les acteurs de système

Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositifs matériels ou autres système) qui interagit directement avec le système (réception d'informations, etc.). Pour notre application, on a trois acteurs : le premier acteur c'est l'administrateur et le deuxième c'est utilisateur de la réception et le troisième c'est l'utilisateur simple.

Acteur	Type Acteur	Descriptions
Administrateur	Acteur Principal	S'authentifier
		Générer le modèle
		Gérer Sous-direction
		Gérer les Services
		Gérer les Tâches
		Gérer le Workflow
		Consulter les bordereaux
		Consulter les historiques des dossiers
		Gérer les Utilisateurs
Utilisateur réception	Acteur Principal	Créer et Gérer les dossiers
		Activer les dossiers et les ajouter sur wf
Utilisateur Simple	Acteur Principal	finir le traitement de dossier
		Gérer les bordereaux
		Gérer les historiques des dossiers

TABLE 2.1 – Les acteurs de système

2.2.3.2 Cas d'utilisation

Le diagramme des cas d'utilisation suivante illustre les fonctionnalités qu'un simple utilisateur du système, également l'administrateur, peut faire. Ce diagramme a été inspiré des spécifications citées ci-dessus :

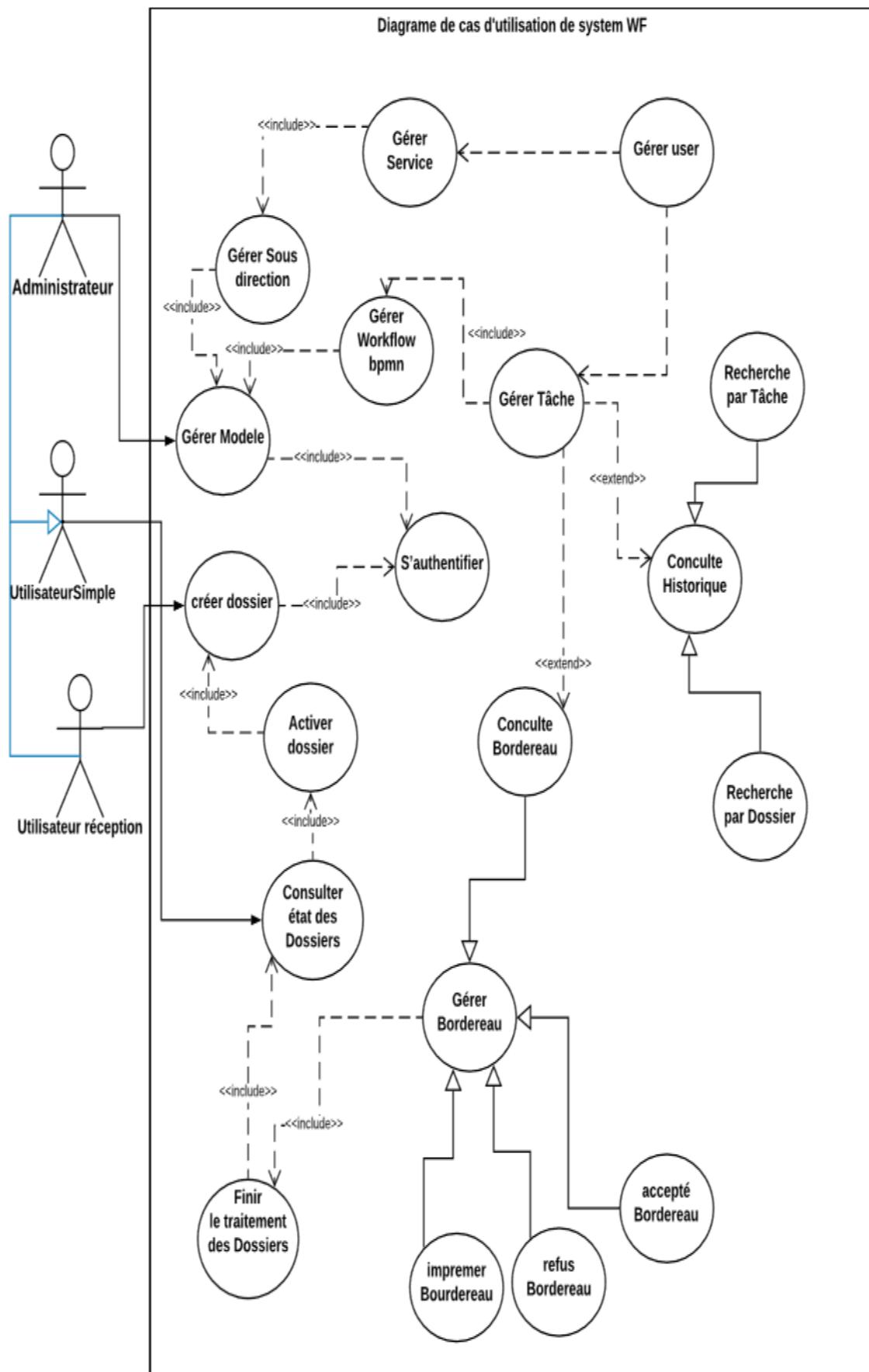


FIGURE 2.1 – Diagramme de Cas d'utilisation

2.2.4 Documentation des cas d'utilisation fonctionnels

Nous détaillerons chaque cas d'utilisation avec une description brève et spécifions les acteurs principaux et les acteurs secondaires, ainsi que les postconditions et les préconditions pour le faire sont la séquence principale et la séquence alternative.

2.2.4.1 S'authentifier

CU : S'authentifier
ID : 1
Description brève : chaque utilisateur doit s'authentifier auprès de l'application afin de pouvoir utiliser les fonctionnalités du système.
Acteurs primaires : utilisateur, administrateur
Acteurs secondaires :
Pré-conditions :
<ul style="list-style-type: none"> - La connexion auprès du serveur d'applications doit être réussite. - L'utilisateur doit être enregistré dans le système.
Enchainement principal : Ce cas d'utilisation commence lorsqu'un utilisateur souhaite accéder à l'application.
<ol style="list-style-type: none"> 1. L'utilisateur saisit le lien de l'application dans barre d'adresse du navigateur. 2. Le serveur répond à l'utilisateur en renvoyant une page d'authentification. 3. L'utilisateur saisit son nom et son mot de passe et en appuyant sur "S'identifier" (Figure3.9). 4. Le serveur vérifie la validité du nom d'utilisateur et du mot de passe. 5. Le serveur envoie une page d'accueil de l'application à l'utilisateur concerné.
Post-conditions : L'utilisateur est connecté.
Enchainement alternatif :
<ul style="list-style-type: none"> — E1 : La page d'authentification n'apparaît pas à l'utilisateur. * L'enchainement démarre après le premier point de l'enchainement principal. * Le serveur envoie un message d'erreur à l'utilisateur. — E2 : Le nom d'utilisateur ou/et le mot de passe ne sont pas validés. * Le serveur envoie un message d'erreur à l'utilisateur. * Le serveur demande à l'utilisateur de ressaisir le nom d'utilisateur et le mot de passe.

2.2.4.2 Créer circuit Workflow

CU : Créer circuit Workflow
ID : 12
Description brève : L'administrateur Crée un circuit de Workflow pour l'enchainement des dossiers
Acteurs primaires : L'administrateur
Acteurs secondaires :
Pré-conditions : L'administrateur doit être connecté
Enchainement principal : Un circuit Workflow est créé pour les dossiers et la création des tâches sur la base de donnée normale.
Post-conditions : Workflow est défié et les tâches à remplir
Enchainement alternatif :

2.2.4.3 Gérer les Services

CU : Gérer les Services
ID : 22
Description brève : L'administrateur peut gérer les services à travers la création, la modification et la suppression d'un ou plusieurs services.
Acteurs primaires : L'administrateur
Acteurs secondaires :
Pré-conditions : L'administrateur doit être connecté
Enchainement principal : L'administrateur crée un nouveau service au système et peut le modifier.
Post-conditions : le service est ajouté sur la base de donnée (BDD)
Enchainement alternatif :

2.2.4.4 Gérer utilisateur

CU : Gérer utilisateur
ID : 12
Description brève : Un administrateur peut gérer des utilisateurs en créant des comptes en spécifiant une tâche pour chacun d'eux.
Acteurs primaires : L'administrateur
Acteurs secondaires :
Pré-conditions : L'administrateur doit être connecté
Enchainement principal : L'administrateur a créé des comptes d'utilisateurs.
Post-conditions : Comptes d'utilisateurs créés
Enchainement alternatif :

2.2.4.5 Consulter Historique

CU : Consulter Historique
ID : 32
Description brève : L'administrateur peut consulter l'historique des dossiers et leur état, ainsi la recherche par dossier ou la recherche par tâche.
Acteurs primaires : L'administrateur
Acteurs secondaires :
Pré-conditions : L'administrateur doit être connecté
Enchainement principal :
-L'administrateur sélectionne le dossier ou la tâche et en appuyant sur rechercher .
-Le système affiche l'état des dossiers.
Post-conditions : Afficher le résultat de la recherche
Enchainement alternatif :

2.2.4.6 Cas d'utilisation par l'utilisateur réception

L'utilisateur de réception a deux rôles principaux : la création et l'activation des dossiers.

2.2.4.7 Créer Dossiers

CU : Créer Dossiers
ID : 2
Description brève : chaque utilisateur peut créer et enregistrer des nouveaux dossiers.
Acteurs primaires : utilisateur de réception ,
Acteurs secondaires : simple utilisateur
Pré-conditions : L'utilisateur doit être connecté au système.
Enchainement principal : Ce cas d'utilisation commence lorsqu'un utilisateur souhaite accéder à l'application.
1. L'utilisateur sélectionne d'ajouter nouveau dossier.
2. système répond à l'utilisateur en affichant un formulaire d'enregistrement.
3. L'utilisateur saisit le code et le nom, le prénom et le numéro de téléphone de propriétaire de dossier.
4. L'utilisateur vérifie le dossier et sélectionne les champs actuels en appuyant sur « Enregistrer ».
5. Le système envoie une page des dossiers.
Post-conditions : le dossier est enregistré et ajouté dans le système.
Enchainement alternatif :
E1 : Le code de dossier n'est pas validé ou/et déjà existé.
o Le serveur envoie un message d'erreur à l'utilisateur.
o Le serveur demande à l'utilisateur de saisir le code de dossier.

2.2.4.8 Activer Dossiers

CU : Activer Dossiers
ID : 3
Description brève :
L'utilisateur peut activer le statut du dossier et commencer au début de la tâche "Ajouter au flux de travail", après avoir complété tous les documents de dossier.
Acteurs primaires : utilisateur réception ,
Acteurs secondaires : simple utilisateur
Pré-conditions : Le dossier doit être enregistré dans le système.
Enchainement principal :
1. L'utilisateur sélectionne la liste des dossiers.
2. système répond à l'utilisateur en affichant les listes des dossiers {"toute la liste", "liste activée ", "liste non activée "}.
3. L'utilisateur sélectionne le dossier
4. le système répond à l'utilisateur en affichant le détail, et après en appuyant sur "Activer".
5. le système vérifie ID "code" de dossier et active leur état
Post-conditions : le dossier est activé dans la première tâche.
Enchainement alternatif :

2.2.4.9 Cas d'utilisation par l'utilisateur simple

l'utilisateur a un rôle principal : consulter l'état du dossier pour finir le traitement créer des bordereaux pour transférer les dossiers (imprimer et refuser ou accepter les bordereaux).

2.2.4.10 Consulter l'état des Dossiers

CU : Consulter l'état des dossiers
ID : 4
Description brève : L'utilisateur peut visualiser le statut des dossiers en cours de traitement en fonction de sa tâche et voir tous les dossiers envoyés par une autre tâche.
Acteurs primaires : utilisateur
Acteurs secondaires :
Pré-conditions :
- Le dossier doit être Activé dans le système. - L'utilisateur doit être connecté au système.
Enchainement principal :
1. L'utilisateur sélectionne le statut des dossiers. 2. Le système répond à l'utilisateur en affichant les listes des dossiers au cours de traitement ou les nouveaux dossiers envoyés
Post-conditions : L'utilisateur visualise le statut des dossiers en fonction de sa tâche.
Enchainement alternatif :

2.2.4.11 Gérer Bordereau

CU : Gérer Bordereau
ID : 6
Description brève : L'utilisateur peut sélectionner une liste des dossiers et gère un bordereau.
Acteurs primaires : utilisateur
Acteurs secondaires :
Pré-conditions :
- Le dossier doit être traité dans la tâche de l'utilisateur. - L'utilisateur doit être connecté au système.
Enchainement principal :
1. L'utilisateur sélectionne la liste des dossiers pour la transférer à la tâche suivante. 2. Le système répond à l'utilisateur et crée un nouveau bordereau et affiche la liste de bordereau correspondant à leur tâche. 3. L'utilisateur peut imprimer et accepter ou refuser le bordereau.
Post-conditions : L'utilisateur crée un bordereau et transfert les dossiers, si le bordereau est accepté par la validation d'un utilisateur de la tâche reçue.
Enchainement alternatif :
E1 : bordereau refusé : * renvoyer à la tâche précédent tous les dossiers.

2.3 Représentation des informations

2.3.1 Diagramme de Classe

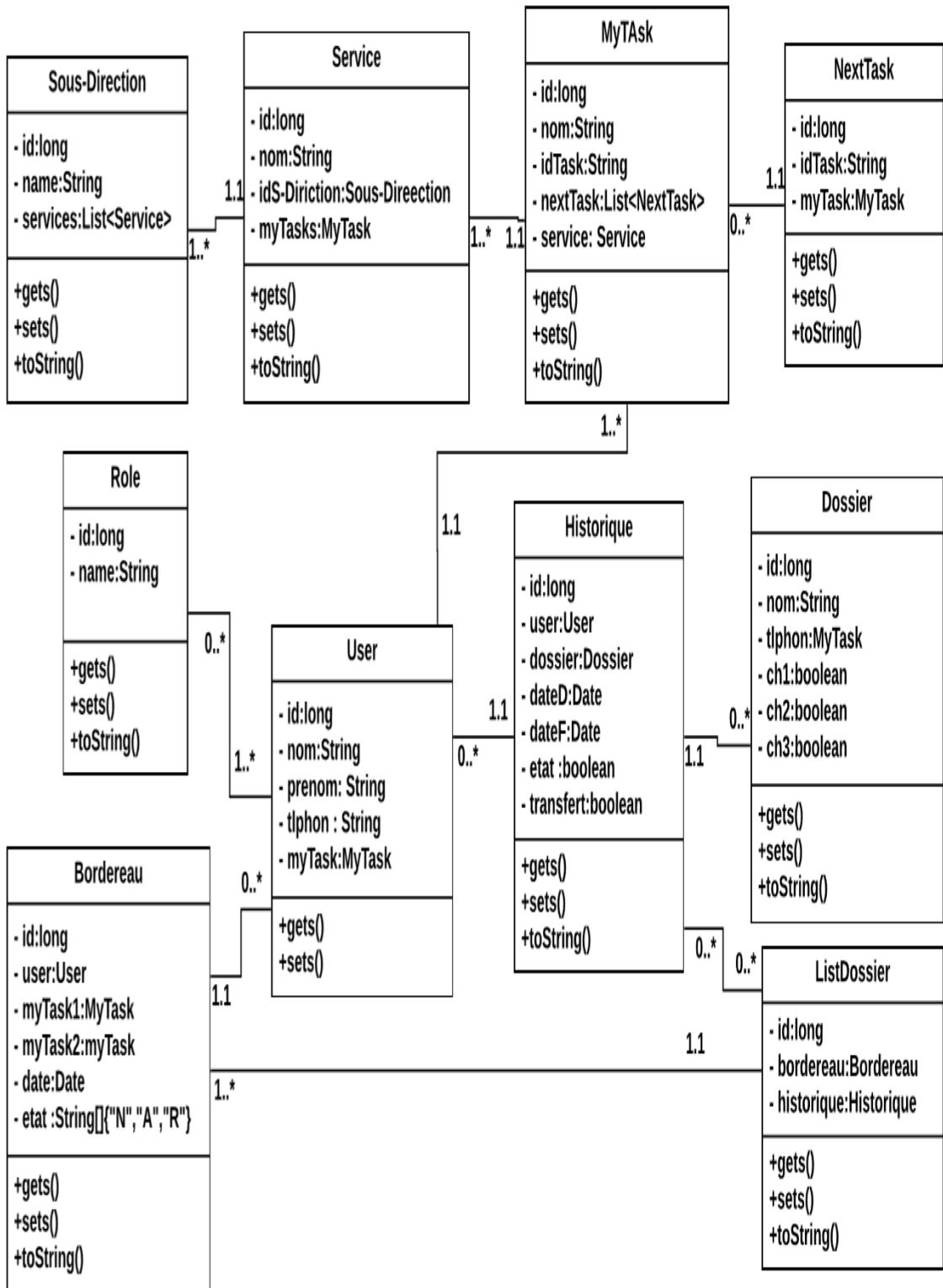


FIGURE 2.2 – Diagramme de Classe

Nous allons définir pour chaque classe ses attributs et leurs types, ainsi que les méthodes qu'elle offre. Dans le but d'alléger le rapport, nous allons cité que les classes essentiels que nous avons conçues.

2.3.1.1 La classe Dossier

Attribut	Type	Méthodes
id	long	getId() et setId()
nom	String	getNom() et setNom()
prenom	String	getPrenom() et setPrenom()
tlphon	String	getTlphon() et setTlphon()
ch1	boolean	getCh1() et setCh1()
ch2	boolean	getCh2() et setCh2()
ch3	boolean	getCh3() et setCh3()

TABLE 2.2 – La classe Dossier

2.3.1.2 La classe Service

Attribut	Type	Méthodes
id	long	getId() et setId()
nom	String	getNom() et setNom()
sous-Direction	Sous-Direction	getSous-Direction() et setSous-Direction()
myTasks	<i>List < MyTask ></i>	getMyTasks() et setMyTasks()

TABLE 2.3 – La classe Service

2.3.1.3 La classe MyTask

Attribut	Type	Méthodes
id	long	getId() et setId()
nom	String	getNom() et setNom()
idTask	String	getIdTask() et setIdTask()
services	Service	getService() et setService()
users	<i>List < User ></i>	getUsers() et setUsers()
nextMyTasks	<i>List < NextMyTask ></i>	getNextMyTasks() et setNextMyTasks()

TABLE 2.4 – La classe MyTask

2.3.1.4 La classe Utilisateur "User"

Attribut	Type	Méthodes
id	long	getId() et setId()
nom	String	getNom() et setNom()
prenom	String	getPrenom() et setPrenom()
tlphon	String	getTlphon() et setTlphon()
myTask	MyTask	getMyTask() et setMyTask()

TABLE 2.5 – La classe Utilisateur

2.3.1.5 La classe NextTask

Attribut	Type	Méthodes
id	long	getId() et setId()
idTask	String	getIdTask() et setIdTask()
myTask	MyTask	getMyTask() et setMyTask()

TABLE 2.6 – La classe NextTask

2.3.1.6 La classe Sous-Direction

Attribut	Type	Méthodes
id	long	getId() et setId()
nom	String	getNom() et setNom()
services	List < Service >	getServices() et setServices()

TABLE 2.7 – La classe Sous-Direction

2.3.1.7 La classe Historique

Attribut	Type	Méthodes
id	long	getId() et setId()
dateD	Date()	getDateD() et setDateD()
dateF	Date()	getDateF() et setDateF()
dossier	Dossier	getDossier() et setDossier()
user	User	getUser() et setUser()
etat	boolean	isEtat() et setEtat()
transfert	boolean	isTransfert() et setTransfert()

TABLE 2.8 – La classe Historique

2.3.1.8 La classe Role

Attribut	Type	Méthodes
id	long	getId() et setId()
nom	Date()	getNom() et setNom()

TABLE 2.9 – La classe Role

2.3.1.9 La classe ListDossier

Attribut	Type	Méthodes
id	long	getId() et setId()
bordereau	Bordereau	getBordereau() et setBordereau()
historique	Historique	getHistorique() et setHistorique()

TABLE 2.10 – La classe ListDossier

2.3.1.10 La classe Bordereau

Attribut	Type	Méthodes
id	long	getId() et setId()
dateD	Date()	getDateD() et setDateD()
user	User	getUser() et setUser()
myTask1	MyTask	getMyTask1() et setMyTask1()
myTask2	MyTask	getMyTask2() et setMyTask2()
etat	String[]	getEtat() et setEtat()
historique	Historique	getHistorique() et setHistorique()

TABLE 2.11 – La classe Bordereau

2.3.2 Diagramme d'activité

À partir de deux diagrammes, cas d'utilisation et de classes, on va exprimer le diagramme d'activité qui montre l'interaction des utilisateurs en fonction de leurs rôles dans le système en général.(figure 2.3)

2.3.2.1 Rôles de l'acteur administrateur :

l'interaction de l'acteur administrateur avec le système, il faut d'abord de connecter au système, puis créer et définir le modèle par la spécification de nom et les champs de dossier pour la génération de codes et lancer un nouveau micro-service et le déployer sur le Cloud par leur nom de service spécifié, ensuite il peut gérer les sous-directions et les services, le modèle de workflow par "BPMN" pour gérer les tâches, gérer les comptes d'utilisateurs, et enfin il consulte les bordereaux qui sont géré par les utilisateurs et de même consulter l'historiques.

2.3.2.2 Rôles de l'acteur utilisateur réception :

L'interaction de l'utilisateur réception, il faut déjà créer ses comptes par l'administrateur pour connecter au système. L'utilisateur peut gérer les dossiers par la création des nouveaux dossiers sur la réception et vérifier les champs de chaque dossier, ensuite il peut activer les dossiers c-à-d lancer le traitement de dossier.

2.3.2.3 Rôles de l'acteur utilisateur simple :

Après l'authentification au système, l'utilisateur se connecter sur leur tâche et il peut voir tous les dossiers en attente qui sont en cours de traitement et consulter les bordereaux qui déjà reçu par une autre tâche, l'utilisateur peuvent finir le traitement de dossier par leur tâche, il peut aussi gérer un nouveau bordereau pour transférer une liste des dossiers vers d'autres tâches et de gérer l'historique des dossiers.

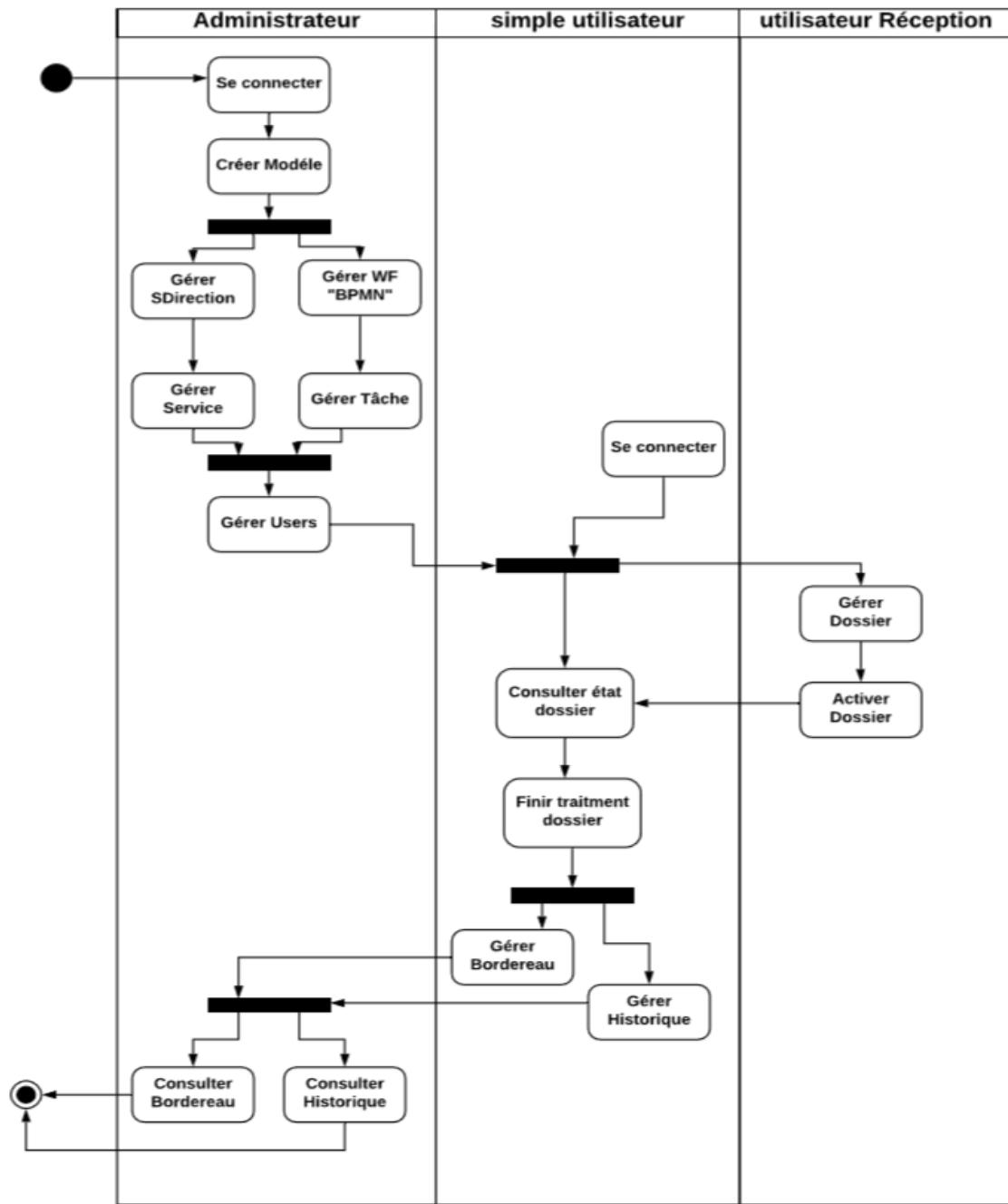


FIGURE 2.3 – Diagramme d'activité qui montre interaction des utilisateurs en fonction de leurs rôles dans le système en général.

2.4 Conclusion

Dans ce chapitre nous avons fait une étude de notre système logiciel en utilisant la modélisation UML après la spécification des besoins.

Dans le chapitre suivant nous allons expliquer la phase de réalisation du système, son intégration et son déploiement sur le cloud par les micro-services, tout en respectant les directives de la conception.

Chapitre 3

Réalisation

Chapitre 3

Réalisation

3.1 Introduction

Une fois la conception validée, il est temps de passer à la réalisation. Nous présentons dans ce chapitre le module réalisé en utilisant des captures d'écran pour montrer ses principales fonctionnalités. Nous commençons par présenter les outils et les technologies utilisées pour le développement, ensuite nous présenterons l'architecture technique de la solution et l'architecture des codes.

3.2 Outils et Technologies utilisées pour le développement

Pour le développement, nous avons utilisé :

3.2.1 Technologies web

3.2.1.1 Technologies web (HTML5, JS et CSS3) :

- **HTML5 (Hypertext Markup Language 5)** : c'est un langage de balisage permettant de décrire la structure et le contenu des pages Web.
- **JS (JavaScript)** : c'est un langage de script orienté objet utilisé pour dynamiser les pages Web et permettre une interaction avec l'utilisateur. Il est exécuté au niveau du navigateur.
- **CSS3 (Cascading Style Sheets)** : c'est un langage permettant de gérer la présentation et la mise en forme des pages web. (Positionnement des éléments, couleurs, tailles et polices, etc...).



Bootstrap Le Framework **Bootstrap** Nous avons utilisé le Framework CSS **Bootstrap3** qui dispose d'une grid pour faciliter la gestion de la mise en forme des pages HTML. Il offre aussi des composants de design basés sur HTML et CSS avec un **Responsive design** (voir la figure 3.1) qui permet un affichage qui s'adapte à la taille de l'écran, que ce soit une tablette, un smartphone ou un ordinateur.



FIGURE 3.1 – Responsive design

3.2.1.2 Thymeleaf

Nous avons utilisé Thymeleaf comme un moteur de template Java pour le traitement et la création de HTML, XML, JavaScript, CSS et du texte.

La bibliothèque est extrêmement extensible et sa capacité naturelle de gabarit permet aux prototypes de prototyper des gabarits, ce qui rend le développement très rapide par rapport aux autres moteurs de gabarits populaires tels que JSP.

3.2.2 Workflow :

Pour l'implémentation de Workflow nous avons utilisé :

- bpmn js
- Camunda BPM

3.2.2.1 bpmn js

bpmn-js est une boîte à outils de rendu et un modélisateur Web BPMN 2.0. pour la modélisation de flux de travaux "Workflow". Il est écrit en JavaScript, intègre les diagrammes BPMN 2.0 dans les navigateurs modernes et ne requiert aucun serveur. Cela facilite son intégration dans n'importe quelle application Web.

3.2.2.2 Camunda BPM

Nous avons utilisé Camunda BPM car est un système de gestion de flux de travaux gratuit écrit en Java qui définit et exécute les processus métier dans BPMN 2.0.

3.2.3 Spring

3.2.3.1 Spring Framework

Spring Framework fournit un modèle complet de programmation et de configuration pour les applications d'entreprise modernes basées sur Java - sur tout type de plate-forme de déploiement.

Un élément clé de Spring est le support infrastructurel au niveau de l'application : Spring met l'accent sur la «plomberie» des applications d'entreprise afin que les équipes puissent se concentrer sur la logique métier au niveau de l'application, sans liens inutiles avec des environnements de déploiement spécifiques. (SPRING 2019)

3.2.3.2 Spring boot

Nous avons utilisé Spring Boot pour faciliter la création d'applications basées sur Spring autonomes et de niveau production que vous pouvez "exécuter".(SPRING 2019)

3.2.3.3 Spring Data

La mission de Spring Data est de fournir un modèle de programmation familier et cohérent, basé sur Spring, pour l'accès aux données tout en conservant les caractéristiques spéciales du magasin de données sous-jacent.

Il facilite l'utilisation des technologies d'accès aux données, des bases de données relationnelles et non relationnelles, des infrastructures de réduction de carte et des services de données en nuage. Il s'agit d'un projet parapluie contenant de nombreux sous-projets spécifiques à une base de données donnée.(SPRING 2019)

3.2.3.4 Spring Security

Pour la partie de sécurité nous avons utilisé **Spring Security** car c'est un cadre d'authentification et de contrôle d'accès puissant et hautement personnalisable. C'est la norme de facto pour la sécurisation des applications basées sur Spring.

Spring Security est une infrastructure qui fournit à la fois l'authentification et l'autorisation aux applications Java. Comme tous les projets Spring, la véritable force de Spring Security réside dans la facilité avec laquelle elle peut être étendue pour répondre aux exigences personnalisées.(SPRING 2019)

3.2.3.5 Spring Cloud

Spring Cloud fournit aux développeurs des outils permettant de créer rapidement certains modèles courants dans les systèmes distribués (gestion de la configuration, découverte de services, disjoncteurs, routage intelligent, micro-proxy, bus de contrôle, jetons à usage unique, verrous globaux, élection des dirigeants, distribution sessions, état du cluster). La coordination des systèmes distribués conduit à des modèles de plaque de chaudière et, grâce à Spring Cloud, les développeurs peuvent rapidement mettre en service des services et des applications mettant en œuvre ces modèles. Ils fonctionneront bien dans n'importe quel environnement distribué, y compris l'ordinateur portable du développeur, les centres de données à nu et les plateformes gérées telles que Cloud Foundry¹.(SPRING 2019)

3.2.3.6 Spring Cloud Netflix

Spring Cloud Netflix fournit des intégrations Netflix pour les applications Spring Boot via la configuration automatique et la liaison à Spring Environment et à d'autres idiomes de modèles de programmation Spring. Quelques annotations simples vous permettent d'activer et de configurer rapidement les modèles courants dans votre application et de créer de grands systèmes distribués avec des composants

1. Cloud Foundry facilite, accélère et simplifie la création, le test, le déploiement et la mise à l'échelle des applications, en offrant un choix de clouds, de cadres de développement et de services d'application. Il s'agit d'un projet open source, disponible via une variété de distributions de cloud privé et d'instances de cloud public.

Netflix testés au combat. Les modèles fournis incluent la découverte de service (Eureka), le disjoncteur (Hystrix), le routage intelligent (Zuul) et l'équilibrage de la charge côté client (ruban).

3.2.4 Micro Service

L'objectif principal de la mise en œuvre des micro-services est de scinder l'application en un service distinct pour chaque fonctionnalité de base et chaque service de l'API. Elle devrait être déployée indépendamment sur le cloud. Nous avons choisi le langage de programmation réactif du projet familial spring.io avec un ensemble de composants pouvant être utilisés pour mettre en œuvre notre modèle d'exploitation. Spring Cloud intègre très bien les composants Netflix dans l'environnement Spring. Il utilise une configuration automatique et une convention de configuration similaire à celle du fonctionnement de Spring Boot.(STEFFENEL 2017)

3.2.4.1 Pourquoi l'architecture de Microservices ?

Nous avons choisi l'architecture de micro-services pour écrire chaque fonctionnalité en tant que service distinct pour les fonctionnalités de base et d'API, ce qui nous aide à réaliser la livraison et l'intégration en continu.(STEFFENEL 2017)

3.2.4.2 Patterns dans l'architecture des microservices

— Api Gatway

1. Choisissez de créer l'application en tant qu'ensemble de micro-services.
2. Décidez comment le client de l'application va interagir avec les micro-services.
3. Avec une application monolithique, il n'y a qu'un seul ensemble de points d'extrémité (généralement répliqués, à charge équilibrée).
4. Dans une architecture de micro-services, chaque micro-service expose cependant un ensemble de points finaux.

— Service registry

1. Le registre de service aide à déterminer l'emplacement des instances de service pour envoyer la demande au correspondant un service
2. Ici, nous avons utilisé Netflix Eureka pour enregistrer un service pouvant être enregistré dans le registre de services. serveur et il peut être identifié par le routeur.

— Service Discovery

1. Dans une application monolithique, les services s'appellent par le biais d'appels de méthode ou de procédure au niveau de la langue.
2. Toutefois, dans une application moderne basée sur des micro-services, elle s'exécute généralement dans des environnements virtualisés où le nombre des instances d'un service et de leurs emplacements change de façon dynamique.
3. Chaque service peut être identifié à l'aide d'un routeur enregistré auprès du serveur de registre de services.

3.2.4.3 Architecture des microservices via les composants Netflix

Nous avons utilisé les composants Netflix pour réaliser les modèles d'architecture de microservices ci-dessus. (OPTISOLBUSINESS 2019).

Operations Component	Spring, Netflix OSS
Service Discovery server	Netflix Eureka
Edge Server	Netflix Zuul
Central configuration server	Spring Cloud Config Server
Dynamic Routing and Load Balancer	Netflix Ribbon
OAuth 2.0 protected API's	Spring Cloud + Spring Security OAuth2
Monitoring	Netflix Hystrix dashboard and turbine

TABLE 3.1 – Architecture des microservices via les composants Netflix

3.2.5 Composantes majeures de Netflix

3.2.5.1 Service Discovery Server



Netflix Eureka permet aux micro-services de s'enregistrer eux-mêmes au moment de leur exécution, tels qu'ils apparaissent dans la structure du système. (OPTISOLBUSINESS 2019)

3.2.5.2 Routage dynamique et équilibrage de charge



Netflix Ribbon peut être utilisé par les consommateurs de services pour rechercher des services au moment de l'exécution. Le ruban utilise les informations disponibles dans Eureka pour localiser les instances de service appropriées. Si plusieurs instances sont trouvées, le Ruban appliquera un équilibrage de charge pour répartir les demandes sur les instances disponibles. Le ruban ne s'exécute pas en tant que service distinct, mais en tant que composant intégré dans chaque consommateur de service.(OPTISOLBUSINESS 2019)

3.2.5.3 Serveur Edge



Zuul est (bien sûr) notre gardien du monde extérieur, ne permettant pas le passage de demandes externes non autorisées. Zulu fournit également un point d'entrée bien connu aux micro-services dans le paysage système. L'utilisation de ports alloués de manière dynamique est pratique pour éviter les conflits de ports et minimiser l'administration, mais elle rend évidemment la tâche plus difficile pour tout consommateur de services donné. Zuul utilise Ribbon pour rechercher les services disponibles et achemine la demande externe vers une instance de service appropriée.(OPTISOLBUSINESS 2019)

3.2.6 Spring Boot et Spring Cloud Netflix OSS – Micro Service Architecture

3.2.6.1 Micro Services avec Spring Boot

Spring Boot est un nouveau framework de l'équipe de Pivotal, conçu pour simplifier le démarrage et le développement d'une nouvelle application Spring. Le cadre adopte une approche de configuration avisée, libérant les développeurs de la nécessité de définir la configuration standard.(OPTISOLBUSINESS 2019)

3.2.6.2 Spring Cloud Netflix

Spring cloud Netflix fournit des intégrations Netflix OSS pour les applications de démarrage printanier via la configuration automatique et la liaison à l'environnement Spring et à d'autres modèles de programmation Spring. Avec quelques annotations simples, nous pouvons rapidement activer et configurer des modèles courants dans une application et construire des systèmes distribués volumineux avec des composants Netflix. De nombreuses fonctionnalités sont disponibles avec le nuage de printemps Netflix. Ici, nous avons répertorié certaines des fonctionnalités communes que nous avons implémentées avec les micro-services avec Spring Boot et Netflix, (OPTISOLBUSINESS 2019)

- **Découverte du service :**

Les instances Eureka peuvent être enregistrées et les clients peuvent Découvrir les exemples à l'aide de haricots à gestion printanière

- **Création de service :** Le serveur Eureka intégré peut être créé avec configuration Java déclarative

- **Configuration Externel :**

Bridge from the Spring Environment (permet aux utilisateurs de configuration des composants Netflix à l'aide de Spring Boot conventions)

- **Routeur et filtre :**

Enregistrement automatique des filtres Zuul, et un simple convention sur l'approche de configuration pour la création de proxy inverse.

3.2.7 Maven

Maven est essentiellement un outil de gestion et de compréhension de projet.

Maven offre des fonctionnalités de :

- Construction, Compilation ;
- Documentation ;
- Rapport ;
- Gestion des dépendances ;
- Gestion des sources ;
- Mise à jour de projet ;
- Déploiement.

3.2.7.1 POM

Le **POM** (Project Object Model) est une façon de décrire, de manière déclarative, un projet au sens de Maven. Cette description est contenue dans le fichier **pom.xml** présent dans le répertoire de base du projet. Le fichier pom.xml contient

donc tous les éléments permettant de gérer le cycle de vie du projet. (DEVELOPPEZ 2019).

Code xml :

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">

<modelVersion>4.0.0</modelVersion>
<groupId>com.mycompany.app</groupId>
<artifactId>my-app</artifactId>
<packaging>jar</packaging>
<version>1.0-SNAPSHOT</version>
<name>Maven_Quick_Start_Archetype</name>
<url>http://maven.apache.org</url>

<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>

</project>
```

- **Project** : c'est la balise racine de tous les fichiers pom.xml.
- **ModelVersion** : cette balise indique la version de POM utilisée. Bien que cette version ne change pas fréquemment, elle est obligatoire afin de garantir la stabilité d'utilisation.
- **GroupId** : cette balise permet d'identifier un groupe qui a créé le projet. Cette clé permet d'organiser et de retrouver plus facilement et rapidement le projet.
- **ArtifactId** : cette balise indique un nom unique utilisé pour nommer les artifacts à construire.
- **Packaging** : type de packaging du projet (ex. : JAR, WAR, EAR, etc.).
- **Url** : adresse du site du projet.
- **Description** : description du projet.
- **Dependencies** : balise permettant de gérer les dépendances.

3.2.7.2 Archetype

Un archetype est un template de projet. Le fait d'utiliser des archetypes pour initialiser un projet permet de gagner du temps et de respecter une certaine convention. (DEVELOPPEZ 2019).

3.2.7.3 Dépendance

Une dépendance est une référence vers un artefact spécifique contenu dans un repository. Cet artefact est nécessaire pour une ou plusieurs phases du cycle de vie du projet. (DEVELOPPEZ 2019).

L'exemple le plus simple est une dépendance sur une bibliothèque jar qui permet d'en utiliser le contenu dans le projet.(DEVELOPPEZ 2019).

3.2.7.4 Artefact

Dans Maven, un artefact est un élément spécifique issu de la construction du logiciel.

Dans Java, les artefacts les plus communs sont des JARs, mais ce peut être aussi un fichier WAR, un EAR, un ZIP, etc.(DEVELOPPEZ 2019).

3.2.7.5 Le groupId/artifactId

Le groupId est l'identifiant du groupe, à l'origine du projet. GroupId suit les mêmes règles de nommage que les packages Java (exemple : fr.masociete.monprojet), et on choisit généralement comme groupId le nom du top package du projet. (DEVELOPPEZ 2019).

L'artifactId est l'identifiant du projet au sein de ce groupe.

L'artifactId est utilisé par défaut pour construire le nom de l'artefact final (exemple : pour un artifactId=monprojet, le nom du fichier jar généré sera monprojet-version.jar).

3.2.7.6 SNAPSHOT

Par convention, une version en cours de développement d'un projet voit son numéro de version suivi d'un -SNAPSHOT.

Ainsi un projet en version 2.0-SNAPSHOT signifie que cette version est une pré-version de la version 2.0, en cours de développement.

Ce concept de SNAPSHOT est particulièrement important pour Maven. En effet, dans la gestion des dépendances, Maven va chercher à mettre à jour les versions SNAPSHOT régulièrement pour prendre en compte les derniers développements.

L'utilisation d'une version SNAPSHOT permet de bénéficier des dernières fonctionnalités d'un projet, mais en contrepartie, cette version peut être (et est) appelée à être modifiée de façon importante, sans aucun préavis. (DEVELOPPEZ 2019).

3.2.7.7 Le repository local et distant

Un repository local est un répertoire sur le poste du développeur permettant de stocker, suivant la même arborescence, tous les artefacts téléchargés depuis le(s) repository distant(s). (DEVELOPPEZ 2019).

Un projet ayant pour POM :

Code xml :

```
<project>
<modelVersion>4.0.0</modelVersion>
<groupId>fr . masociete</groupId>
<artifactId>monprojet</artifactId>
<version>1.0</version>
</project>
```

3.3 Crédit d'un service cloud par spring et micro service :

3.3.1 Outils et Versions

- Spring Boot Version : 2.0.1
- Spring Cloud Version 1.2.4
- Java Version 1.8.0
- Maven Version 3.6.1
- Tu peux utiliser l'éditeur eclipse STS "eclipse Spring Tool Suite" ou IntelliJ IDEA (votre choix).

Nous allons donc créer les microservices suivants :

3.3.1.1 Service principal :

Service principal, qui offre par exemple une API REST pour la liste des utilisateurs du système. (voir la Figure 2.2 dans le chapitre 2 qui représente le diagramme de classe).

3.3.1.2 Config Service :

Service de configuration, dont le rôle est de centraliser les fichiers de configuration des différents microservices dans un endroit unique.

3.3.1.3 Proxy Service :

Passerelle se chargeant du routage d'une requête vers l'une des instances d'un service, de manière à gérer automatiquement la distribution de charge.

3.3.1.4 Discovery Service :

Service permettant l'enregistrement des instances de services en vue d'être découvertes par d'autres services.

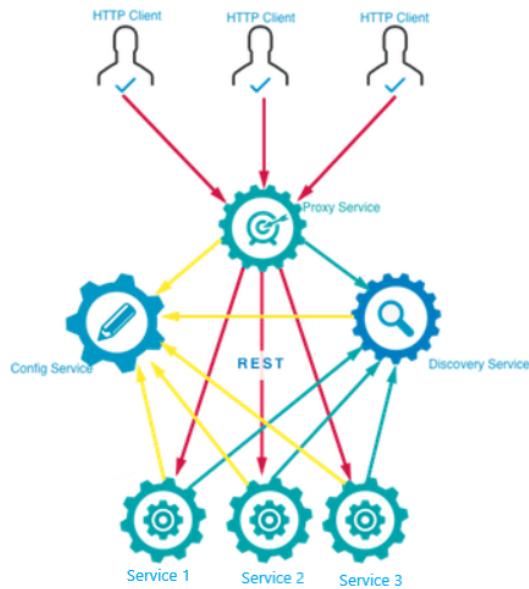


FIGURE 3.2 – Architecture Spring cloud et Netflix par des microservices

3.3.2 Crédation des Microservices

3.3.2.1 Microservice Service 1

Nous commençons par la création du service principal : Service 1.

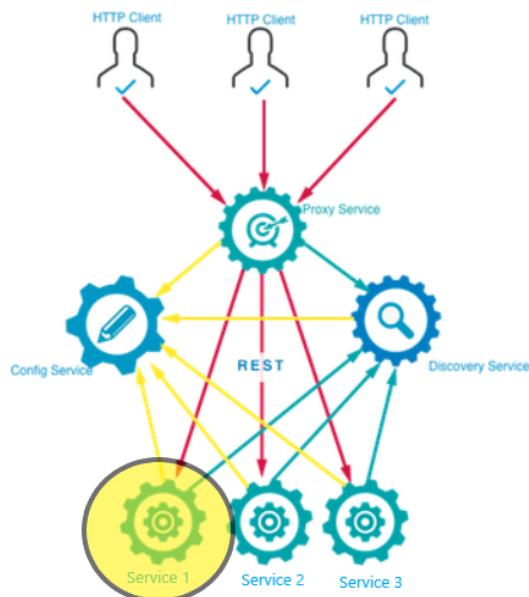


FIGURE 3.3 – Microservice Service 1

Chaque microservice aura la forme d'un projet Spring. Pour créer rapidement et facilement un projet Spring avec toutes les dépendances nécessaires, Spring Boot fournit Spring Initialisé, il faut accéder au site start.spring.io, et créer un projet avec les caractéristiques suivantes :

1. Projet Maven avec Java et Spring Boot version 2.0.1

2. Group : tn.insat.tpmicro

3. Artifact : User-service

4. Dépendances :

- Web
- Rest Repositories
- JPA : Java Persistence API
- Mysql : base de données pour le stockage
- Actuator : pour le monitoring et la gestion de l'application
- Eureka Discovery : pour l'intégration avec le Discovery Service
- Config Client : pour l'intégration avec le Config Service

Suivre ensuite les étapes suivantes pour créer le microservice UserService :

- Ouvrir le projet téléchargé avec votre éditeur.
- Sous le répertoire src/main/java et dans le package com.example.entitis, créer la classe User (**voir annexe A.1**) .

5. Cette classe est annotée avec JPA, pour stocker ensuite les objets user dans la base de données mysql grâce à Spring Data. Pour cela, créer l'interface UserRepository dans le même package (**voir annexe A.2**) .

6. Pour insérer les objets dans la base, nous utiliserons l'objet Stream. Pour cela, nous allons créer la classe DummyDataCLR (**voir annexe A.3**).

7. Nous remarquons ici que le UserRepository sera instancié automatiquement grâce au mécanisme d'injection de dépendances, utilisé par Spring.

Lancer la classe principale. Une base de données MySQL sera créée et le CommandLineRunner se chargera de lui injecter les données.

Attention 3.3.1. Prenez soin d'utiliser JDK 8 !

8. Pour exécuter votre application :

* Créer une configuration mvn package par l'éditeur

* Lancer ensuite la configuration Spring Boot UserServiceApplication

3.3.2.2 Microservice ConfigService

Dans une architecture microservices, plusieurs services s'exécutent en même temps, sur des processus différents, avec chacun sa propre configuration et ses propres paramètres. Spring Cloud Config fournit un support côté serveur et côté client pour externaliser les configurations dans un système distribué. Grâce au service de configuration, il est possible d'avoir un endroit centralisé pour gérer les propriétés de chacun de ces services

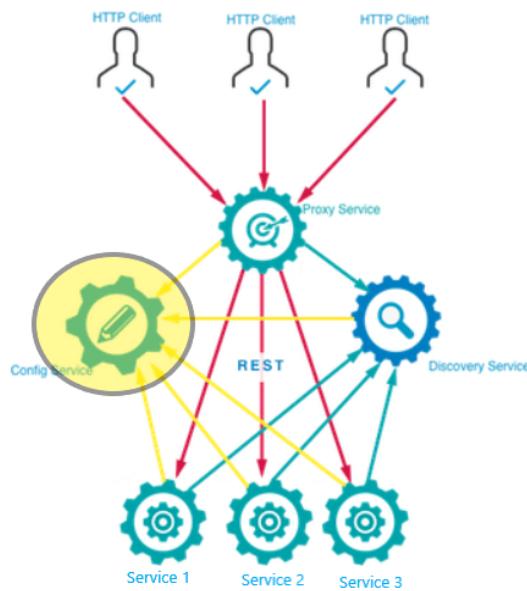


FIGURE 3.4 – Microservice ConfigService

Pour cela :

1. Commencer par créer un service ConfigService dans Spring Initializr, avec les dépendances appropriées, comme indiqué sur la figure suivante :
2. Ouvrir le projet dans une autre instance d’IntelliJ IDEA.
3. Pour exposer un service de configuration, utiliser l’annotation `@EnableConfigServer` pour la classe `ConfigServiceApplication`, (Voir annexe B.1)
4. Pour paramétriser ce service de configuration, ajouter dans son fichier `application.properties` les valeurs suivantes :

```
server.port=8888
spring.cloud.config.server.git.uri=file:./src/main/
resources/myConfig
```

Ceci indique que le service de configuration sera lancé sur le port 8888 et que le répertoire contenant les fichiers de configuration se trouve dans le répertoire `src/main/resources/myConfig`. Il suffit maintenant de créer ce répertoire.

5. Créer le répertoire `myConfig` à l’arborescence `src/main/resources`
6. Créer dans ce répertoire le fichier `application.properties` dans lequel vous insérez l’instruction suivante :

```
global=xxxxx
```

Ce fichier sera partagé entre tous les microservices utilisant ce service de configuration.

7. Le répertoire de configuration doit être un répertoire git. Pour cela :
 - Ouvrir le terminal avec IntelliJ et naviguer vers ce répertoire.
 - Initialiser votre répertoire : `git init`
 - Créer une entrée racine dans le repository : `git add .`

— Faire un commit : **git commit -m "add ."**

Revenir vers le projet ProductService et ajouter dans le fichier de configuration application.properties :

```
spring.application.name = product-service
spring.cloud.config.uri = http://localhost:8888
```

Redémarrer vos services. Pour consulter le service de configuration, aller à **http://localhost:8888/product-service/master**.

Vous verrez le fichier JSON (**Voir annexe B.2**)

Comme le fichier application.properties contient toutes les propriétés partagées des différents microservices, nous aurons besoin d'autres fichiers pour les propriétés spécifiques à un microservice. Pour cela :

8. Créer dans le répertoire myConfig un fichier product-service.properties pour le service ProductService.
9. Ajouter les propriétés de votre service, à savoir, par exemple :

```
me=Djamel.Zerroukki@jimmi.fr
```

Relancer le microservice de configuration. En consultant l'url **http://localhost:8888/product-service/master**, nous remarquons l'ajout de la nouvelle propriété. (**Voir annexe B.3**)

Nous allons maintenant définir un appel REST à cette propriété. Pour cela :

10. Créer la classe ProductRestService dans le projet product-service. (**Voir annexe B.4**)
11. Redémarrer l'instance du service, puis appeler dans votre navigateur le service en tapant : **http://localhost:8080/messages**.

3.3.2.3 Microservice DiscoveryService

Pour éviter un couplage fort entre microservices, il est fortement recommandé d'utiliser un service de découverte qui permet d'enregistrer les propriétés des différents services et d'éviter ainsi d'avoir à appeler un service directement. Au lieu de cela, le service de découverte fournira dynamiquement les informations nécessaires, ce qui permet d'assurer l'élasticité et la dynamique propres à une architecture microservices.

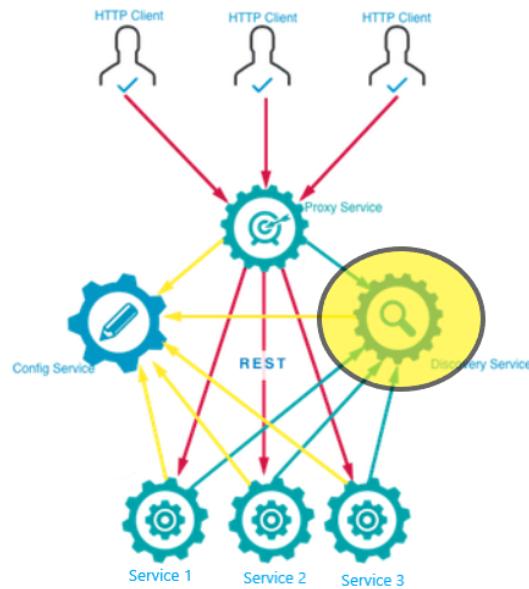


FIGURE 3.5 – Microservice DiscoveryService

Pour réaliser cela, Netflix offre le service **Eureka Service Registration and Discovery**, que nous allons utiliser dans notre application.

1. Revenir à Spring Initializr et créer un nouveau projet Spring Boot intitulé discovery-service avec les dépendances Eureka Server et Config Client.
2. Lancer le projet avec votre éditeur.
3. Dans la classe DiscoveryServiceApplication, ajouter l'annotation **@EnableEurekaServer**. (Voir annexe C)
4. Ajouter les propriétés suivantes dans son fichier application.properties.

```
spring.application.name=discovery-service
spring.cloud.config.uri=http://localhost:8888
```

5. Dans le projet config-service, créer un fichier discovery-service.properties sous le répertoire myConfig.
6. Ajouter les propriétés suivantes pour (1) définir le port par défaut du service de découverte et (2) empêcher un auto-enregistrement du service Eureka.

```
server.port = 8761
eureka.client.fetch-registry = false
eureka.client.register-with-eureka = false
```

Pour consulter le service Eureka, aller à <http://localhost:8761>, l'interface suivante s'affiche :

3.3.2.4 Microservice ProxyService

L'architecture microservice, en fournissant un ensemble de services indépendants et faiblement couplés, se trouve confrontée au challenge de fournir une interface unifiée pour les consommateurs, de manière à ce qu'ils ne voient pas la décomposition à

faible granularité de vos services. C'est pour cela que l'utilisation d'un service proxy, responsable du routage des requêtes et de la répartition de charge, est important.

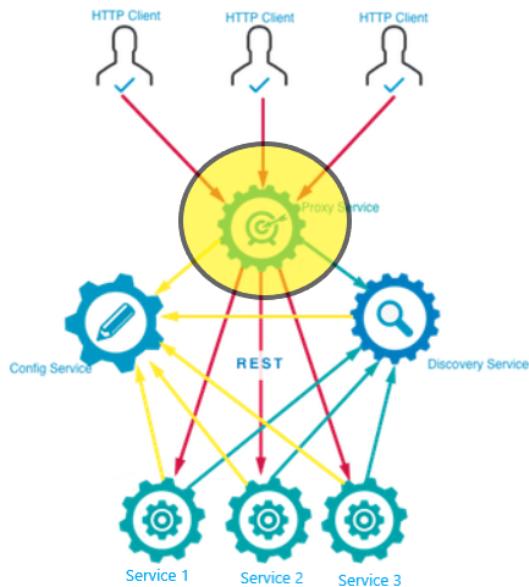


FIGURE 3.6 – Microservice ProxyService

Netflix offre le service **Zuul** pour réaliser cela. Pour créer votre microservice Proxy :

1. Aller à Spring Initializr.
2. Créer le projet proxy-service avec les dépendances suivantes : (**Zuul**, **Web**, **HATEOAS**, **Actuator**, **Config Client** et **Eureka Discovery**).
3. Ouvrir le service avec votre IDEA.
4. Ajouter à la classe `ProxyServiceApplication` l'annotation `@EnableZuulProxy`, ainsi que `@EnableDiscoveryClient` pour que le proxy soit également enregistré dans le service ce découverte.(Voir annexe D)
5. Ajouter les propriétés `spring.application.name` et `spring.cloud.config.uri` dans le fichier `application.properties` du service proxy.
6. Créer le fichier `proxy-service.properties` dans le répertoire `myConfig` du service de configuration, dans lequel vous allez fixer le port du service proxy à 9999.

En lançant le service Proxy, vous remarquerez qu'il est rajouté dans Eureka.

3.3.3 Consulter les services via le proxy

le diagramme de séquence de la figure 3.7 exprime la consultation des services via le proxy

1. Un client HTTP envoie la requête HTTP vers le proxy qui contient le nom de micro service.
2. Le service proxy connaît le nom de micro service, il va demander au service de registration l'emplacement de la machine où se trouve ce micro-service.

3. Le service de registration va retourner l'adresse IP et le port s'il a une instance ou une liste s'il a plusieurs instances.
4. Le service proxy va choisir une instance et va envoyer la requête.
5. Le service demandé va faire le traitement ensuite envoie le résultat vers le proxy qui va le transmettre au client.

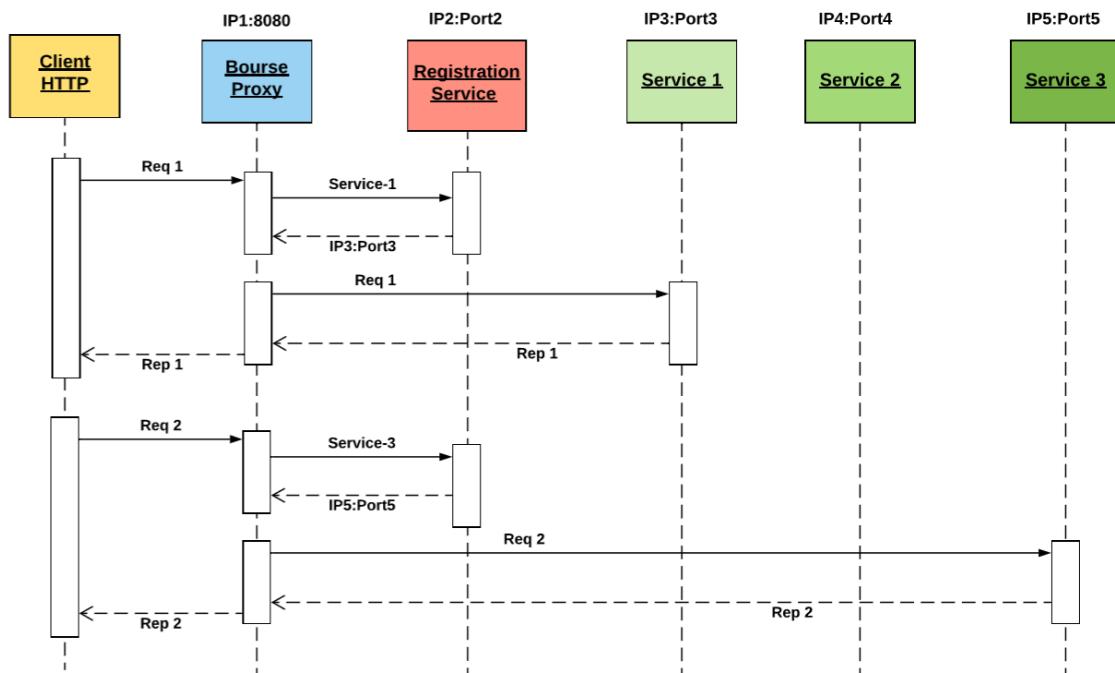


FIGURE 3.7 – Diagramme séquence représenté comment Consulter les services via le proxy

3.4 Description de l'architecture de notre système

Nous avons instanciés le workflow dans plusieurs dimensions des cas (voir le paragraphe 1.3.4.2).

Nous avons développé trois services pour l'orchestration des Micro-Service à savoir service proxy, service configuration et service registration comme le montre la figure 3.8.

De plus, nous avons développé un master service qui permettre d'instancier le workflow pour diverses organisations. Autrement dit, le fournisseur de services offre ses services via le master service.

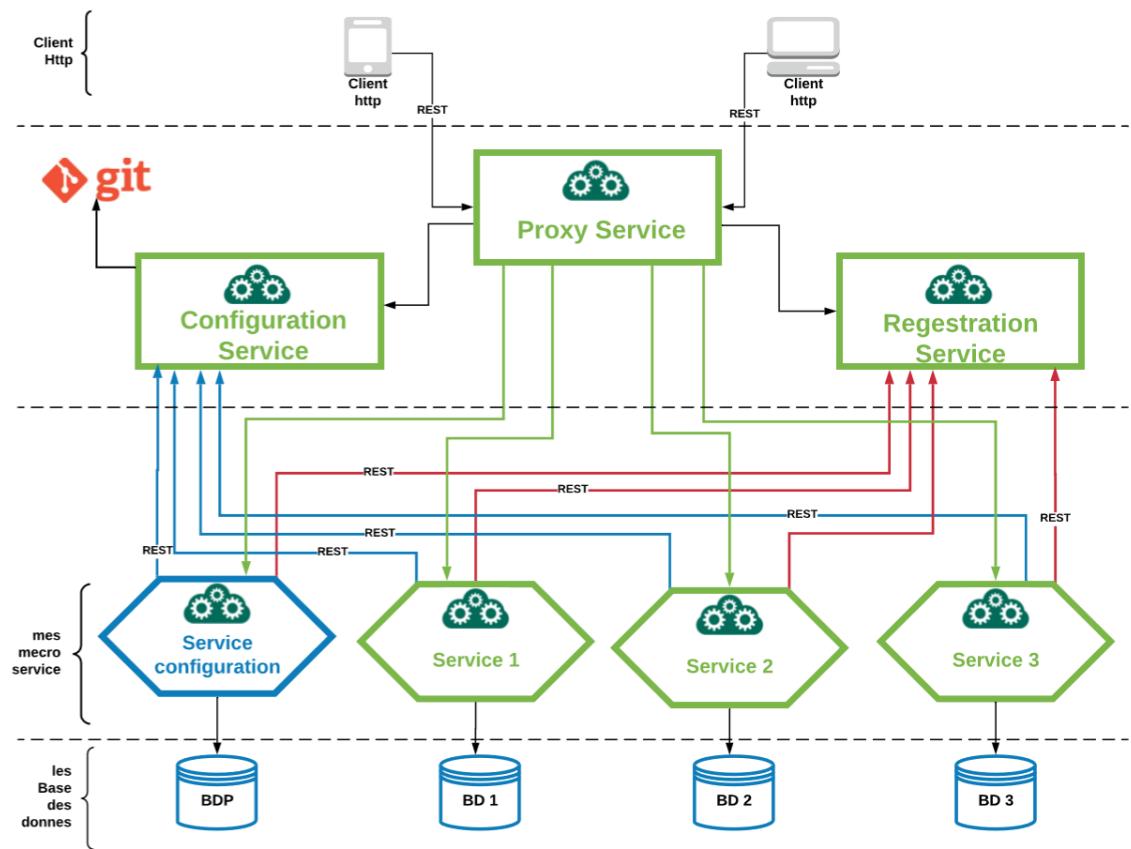


FIGURE 3.8 – Notre architecture de système sur l'environnement Spring cloud et micro-service

Suite à la demande de la CNR nous avons instanciés notre workflow pour cette organisation et nous avons amélioré notre application suivent leur cahier des charges (Voir **annexe E** Figure E.1 qui représente L'organigramme de la CNR).

3.4.1 Obstacles

Nous avons essayé d'ouvrir un compte auprès des fournisseurs de services cloud cités dans la section 1.2.6 , pour déployer les services que nous avons développés, malheureusement sont tous payants.

3.5 Présentation de système

3.5.1 Interface de l'administrateur

3.5.1.1 Authentification

Cette interface est la fenêtre de bienvenue. Pour améliorer la sécurité du système, il est nécessaire de vérifier la disponibilité du compte d'utilisateur et le mot de passe est correct à ce niveau.

The screenshot shows a login interface titled "Login Form". It features two input fields: "Username" and "Password", each with a small icon (envelope and briefcase respectively) to its right. Below these is a checkbox labeled "Remember me". To the right of the password field is a blue "Log in" button. At the bottom of the form, there is a light blue horizontal bar containing the text "You have been disconnected." in a small font.

FIGURE 3.9 – Authentification au système

3.5.1.2 Crédation d'un circuit workflow

L'administrateur peut créer un circuit de n'importe quel dossier via une interface graphique simple et utile.

Example 3.5.1. Crédation d'un workflow pour un service de la CNR comme illustré à la figure 3.10.

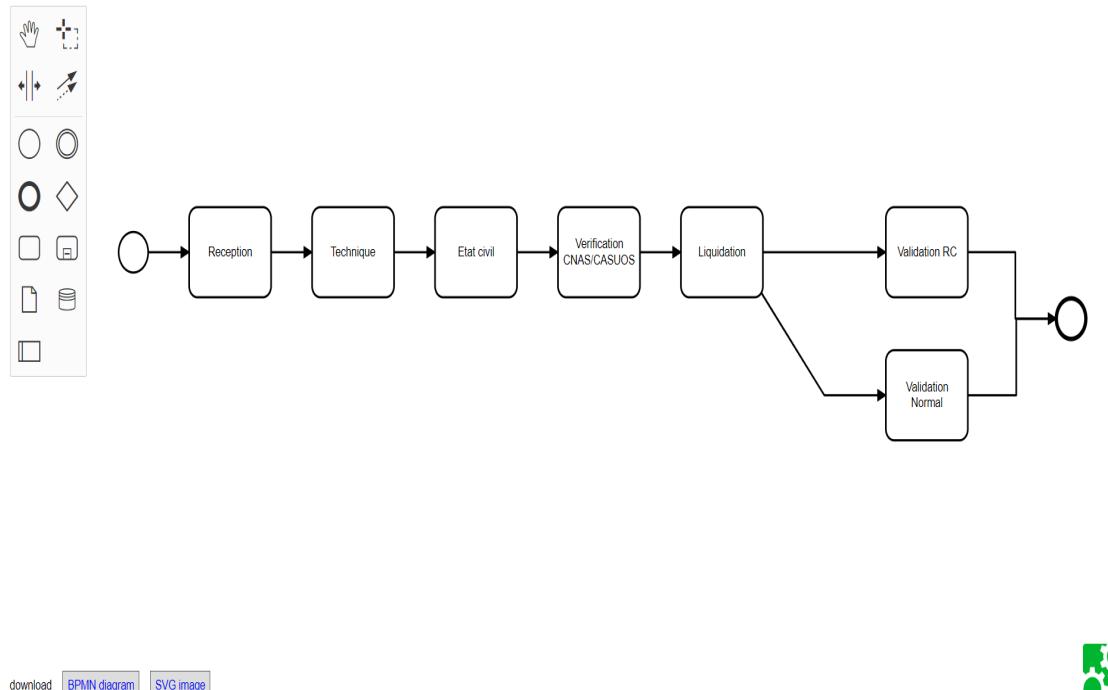


FIGURE 3.10 – Un workflow pour un service de la CNR

3.5.1.3 Gestion des Tâches

L'administrateur peut gérer des tâches en modifiant des informations telles que le service et le temps de traitement. Il peut également afficher l'état des dossiers dans chaque tâche.

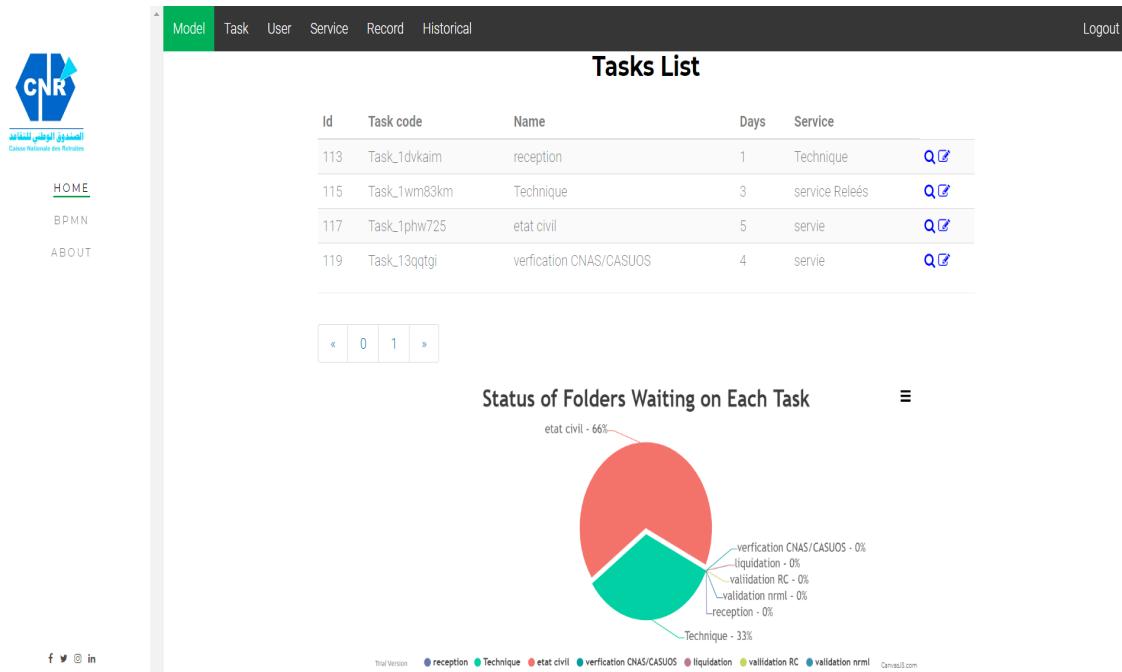


FIGURE 3.11 – Gestion des Tâches

3.5.1.4 Consultation l'historique par Tâche

L'administrateur peut consulter le résultat de la recherche par tâche et l'état actuel des dossiers.

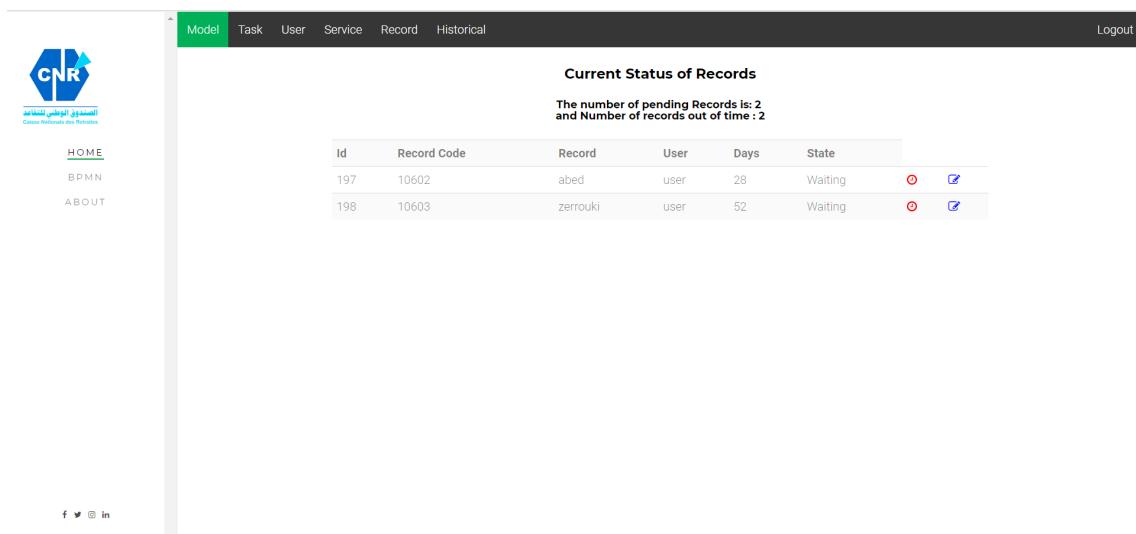


FIGURE 3.12 – Consultation Historique par Tâche

3.5.1.5 Gestion des Sous-Directions et des Services

L'administrateur peut gérer des sous-directions et des services en ajoutant et en modifiant leurs informations.

The screenshot shows a web application for managing services. On the left, there's a sidebar with links for HOME, BPMN, and ABOUT. The main area has a title 'Services' and a sub-section 'Add New'. A modal dialog is open, titled 'Service', containing fields for 'Name' (controleur) and 'Sub-Directorate'. Below the modal is a table listing existing services with columns for Id, Name, Sub-Directorate, and edit/delete icons. At the bottom of the page are social media sharing icons.

Id	Name	Sub-Directorate
67	Technique	presions
68	service Releés	reconstitution
70	liquidation	presions
79	servie	presions
80	new service	ADMGX

FIGURE 3.13 – Gestion des Sous-Directions et des Services

3.5.1.6 Gestion des Utilisateurs

L'administrateur peut gérer les comptes d'utilisateurs en ajoutant, modifiant ou supprimant leurs informations.

The screenshot shows a web application for managing users. The top navigation bar includes Model, Task, User, Service, Record, Historical, and Logout. The main area has a title 'Users List' with an 'Add +' button. Below is a table listing users with columns for Id, Name, phone, Task, Service, and edit/delete icons. At the bottom, there's a navigation bar with page numbers (0, 1, 2, 3, 4) and a footer with social media icons.

Id	Name	phone	Task	Service
126	admin	0669706401	reception	Technique
127	user	0774952451	Technique	service Releés
128	khaled	0778807788	etat civil	servie
129	rabah	0778807788	verfication CNAS/CASUOS	servie

FIGURE 3.14 – Gestion des Utilisateurs

3.5.1.7 Consultation des Dossiers

Cette page permet à l'administrateur de voir tous les dossiers en cours de traitement et son statut par la recherche.

Id	Name	FirstName	Phone
10602	abed	fatma	0789456123
10603	zerrouki	nasrEddine	0774952451
10652	Arrousi	ilyas	0669706402
10702	bouziyan	khaled	0778807701
10752	kliklik	djamel	0669706401
10802	kadi	nono	0778807788
10852	chayma	chayma	0669706400
10853	habiba	habiba	0669706403
10902	abed	ali	0778807788
10952	zerrouki	ali	0669706401

FIGURE 3.15 – Consultation des Dossiers

3.5.1.8 Consultation l'historique par dossier

Le résultat de la recherche est affiché par dossier comme suit :(figure 3.16)

Id	Record	User	State	Task	Service	Days
189	abed	admin	true	reception	Technique	1
196	abed	user	true	Technique	service Reléés	2
197	abed	user	false	etat civil	servie	28

The total time taken so far is 31 days

Gantt chart for record processing

Tasks

reception - 0D to 10D

Technique - 10D to 30D

etat civil - 30D to 310D

Date by day

FIGURE 3.16 – Consultation Historique par dossier

3.5.2 l'interfaces de l'Utilisateur

3.5.2.1 Accueil

La page d'accueil pour des utilisateurs de système "les employés".



FIGURE 3.17 – La page d'accueil

3.5.2.2 Ajouter un dossier

L'utilisateur de réception crée des dossiers sur cette interface en remplir toutes les informations et en vérifiant les champs.

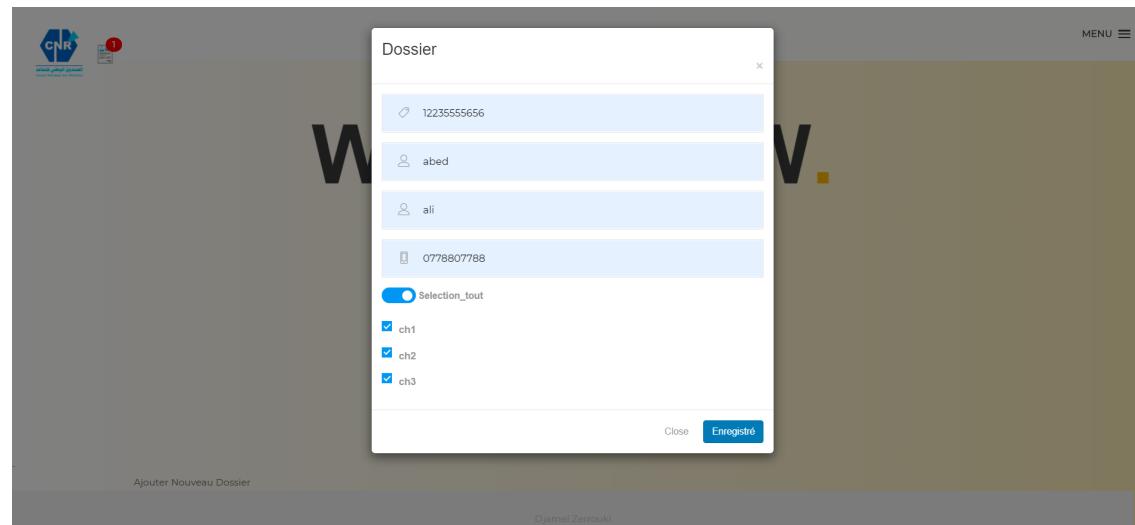


FIGURE 3.18 – Ajouter un dossier

3.5.2.3 Liste des dossier

Affichez et filtrez les dossiers de la manière suivante : "En attente", "Traitement en cours" ou "Tous les fichiers".

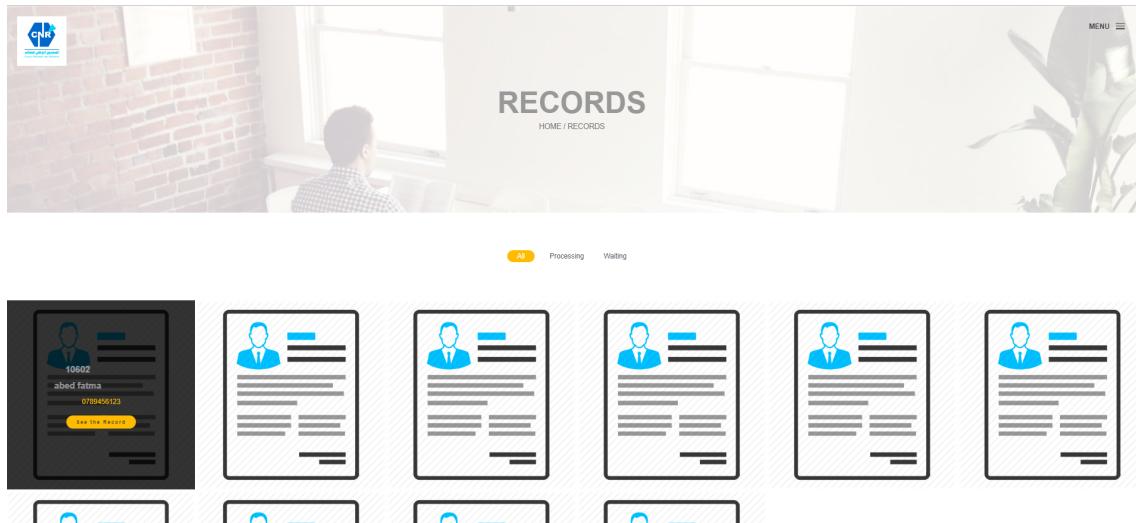


FIGURE 3.19 – Liste des dossier

3.5.2.4 Gérer les bordereaux

L'utilisateur peut sélectionner liste des dossiers et créer un bordereau pour transformer et la gérer ("Accepter", "Refuses", "Imprimer").

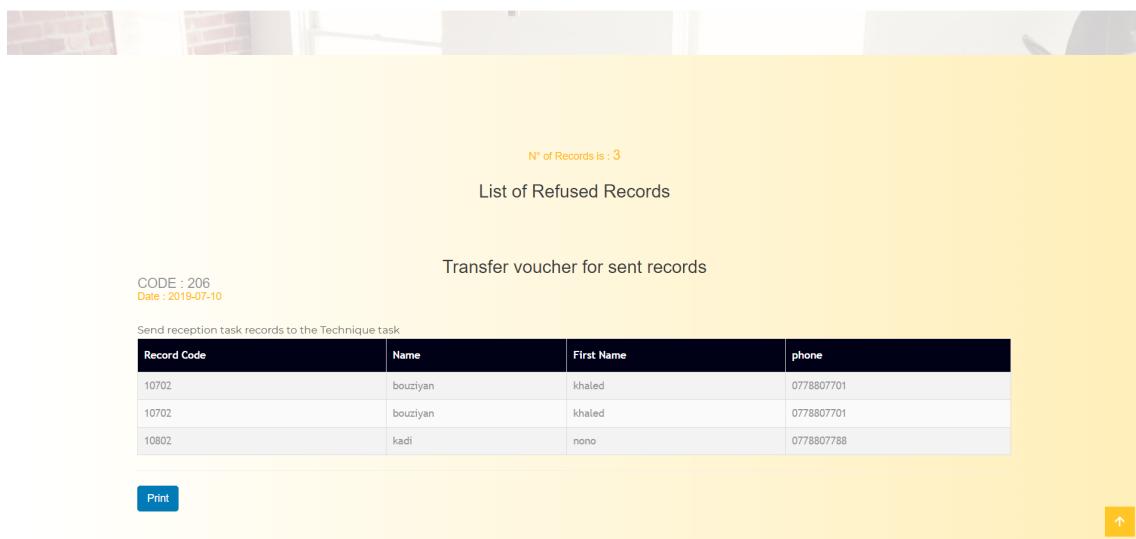


FIGURE 3.20 – Gérer les bordereaux

3.6 Conclusion

À travers ce chapitre, nous avons exposé l'architecture de la plateforme, les différents outils et technologies utilisées pour le développement d'un service cloud. Par la suite, nous avons présenté la CNR comme dimension de cas. Enfin, nous avons présenté un aperçu du système que nous avons réalisé.

Conclusion générale

Dans ce projet de fin d'étude, nous sommes intéressés à améliorer le processus de travail des entreprises et des organisations modernes en Algérie, par le développement d'un système qui fournit des services permettant d'accélérer, organiser et contrôler les activités ainsi le travail de ces entreprises et organisations.

L'objectif de ce projet est la conception et la mise en place d'un service Cloud qui fournit aux organisations un système Workflow administrative.

Afin d'atteindre cet objectif, nous avons commencé à définir l'approche en matière de direction de projet : il était tout d'abord nécessaire d'établir un état des connaissances dans le Cloud et dans le Workflow, puis nous avons procédé à une analyse des besoins, une phase de conception et finalement l'implémentation de la solution.

La meilleure solution consiste à répondre aux besoins spécifiques en fournissant des services de flux de travail flexibles, permettant aux organisations d'automatiser les processus métier pour formaliser, normaliser et accélérer le traitement des documents et autres tâches.

Perspective

Néanmoins, certaines perspectives peuvent être envisagées, à savoir :

- Déployer les services en ligne pour sa mise à disposition au grand public.
- La notification des clients par une application mobile ou par SMS pour suivre leurs dossiers.

Bibliographie

- ADER, M. (1996). *management collectif de l'information.*
- BELOUNNAR, Saliha (2011). "Une approche formelle pour l'adaptabilité d'un agent". Thèse de doct. Université Mohamed Khider Biskra.
- BOUCHHIMA, Aimen (2006). "Modélisation du logiciel embarqué à différents niveaux d'abstraction en vue de la validation et la synthèse des systèmes monopuces". Thèse de doct. Institut National Polytechnique de Grenoble-INPG.
- BUYYA, R. et VENUGOPAL (2004). *The Gridbus Toolkit for Service Oriented Grid and Utility computing.* Seoul, Korea : 1st IEEE International Workshop on Grid Economics and Business Models (GECON 2004), p. 19-36.
- COMPUTING, Cloud (2019). *Convergence des différentes avancées menant à l'avènement du cloud computing.* URL : https://www.google.com/search?q=Convergence+des+diff%C3%A9rentes+avanc%C3%A9es+menant+%C3%A0+l%27av%C3%A8nement+du+cloud+computing.&rlz=1C1CHBD_deDZ776DZ776&tbo=isch&source=lnms&sa=X&ved=0ahUKEwiSg7H6kZHjAhX5QxUIHWOpDw8Q_AUICygC&biw=767&bih=712&dpr=1.25#imgrc=XezSbm7RA6Tj9M:
- COURTOIS, T. (1996). "Workflow : la gestion globale des processus de l'entreprise." In : *Logiciels and Systèmes.*
- DEVELOPPEZ (2019). *FAQ Maven.* URL : <https://java.developpez.com/faq/maven/?pages=Terminologie-et-documentation#Qu-est-ce-que-Maven>.
- EPSI (2019). *PALMARÈS 2019 DES FOURNISSEURS DU CLOUD.* URL : <http://www.epsi.fr/fournisseurs-cloud-2019/>.
- FOSTER, I. et KESSELMAN (1999). "blueprint for a new computing infrastructure". In : *Morgan Kaufmann Publishers Inc.*
- GARFINKEL, S. et ABELSON (1999). *35 Years of the Laboratory for Computer Science.* Cambridge, MA, USA : Mit. MIT Press, p. 235-246.
- GEELAN (2019). *Twenty One Experts Define Cloud Computing.* URL : <http://virtualization.sys-con.com/node/612375>.
- GEORGAKOPOULOS, D. (1995). *workflow management from process modeling to workflow automation infrastructure.* Distributed et Parallel Databases, p. 119-153.
- INCONCERT (1997). *InConcert Specifications Document.* URL : <http://www.inconcert.com>.
- MG (2019). *Unified Modeling Language.* URL : www.uml.org.
- MCREADY, S. (1992). *There is more than one kind of Workflow software.* Computerworld, p. 86-90.
- MELL (2011). "The NIST Definition of Cloud Computing". In : *Ried.*
- MERIEM, Meddeber (2012). "Placement Dynamique de Tâches dans une Grille de Calcul". Thèse de doct. Université d'Oran.

-
- OPTISOLBUSINESS, OB. (2019). *MicroServices Architecture – Spring Boot And Netflix Infrastructure*. URL : <https://www.optisolbusiness.com/insight/micro-services-architecture-spring-boot-and-netflix-infrastructure>.
- RIED S., Lisserman H. Matzke-P. Bartels A. et KISKER (2011). “Algorithmic Graph Theory and Perfect Graphs”. In : *Ried*. URL : http://www.forrester.com/rb/Research/sizing_cloud/q/id/58161/t/2..
- RIMAL (2009). *cloud computing systems*, p. 44-51.
- SADIQ, S.W. (1999). *Workflows in Dynamic Environments Can they be manage*. Computer science et Electrical Engineering, p. 165-176.
- SBAÏ, Zohra (2010). “Contribution à la modélisation et à la vérification de processus workflow”. Thèse de doct. Conservatoire national des arts et métiers-CNAM.
- SCHEER A.W., Borowsky S. Klabunde S. et A. TRAUT. (1997). *Flexible industrial applications through model-based Workflows*. Torino Italy : he International Conference on Enterprise Integration et Modeling Technology, p. 439-448.
- SPRING (2019). *spring projects documentation*. URL : <https://spring.io/projects/>.
- STEFFENEL, Luiz Angelo (2017). “Contributions à la Gestion de l’Hétérogénéité dans les Environnements Distribués et Pervasifs”. Thèse de doct. Université de Reims Champagne Ardenne.
- VANDERAALST, W.M.P. (1997). *Verification of Workflow Nets. Application and Theory of Petri Nets, Lecture Notes in Computer Science Springer*. P. Azema et G. Balbo, editors, p. 407-426.
- VOORHOEVE, M. (1997). “Ad-hoc Workflow : Problems and Solutions”. In : *the 8 th DEXA Workshop Conference*, p. 36-41.
- VOUK (2008). *cloud computing*. Computing et Information Technology, p. 235-246.
- WANG (2008). *Scientific cloud computing*. Washington, DC, USA : the 10th IEEE Int. Conf on High Performance computing et Communications, p. 825-830.
- WFMC (1999). “Terminology and Glossary”. In : *The Workflow Management Coalition Specification*, p. 1-64. URL : https://www.wfmc.org/docs/TC-1011_term_glossary_v3.pdf.
- YASSA, Sonia (2014). “Allocation optimale multicontraintes des workflows aux ressources d’un environnement Cloud Computing”. Thèse de doct. Cergy-Pontoise.
- ZACHAREWICZ, Gregory (2006). “Un environnement G-DEVS/HLA : Application à la modélisation et simulation distribuée de workflow”. Thèse de doct. Université de droit, d’économie et des sciences-Aix-Marseille III.
- RDP (2008). *Etat de l’art sur les deux méthodes de modélisation de Workflow*. URL : <http://www.univ-bejaia.dz/dspace/bitstream/handle/123456789/9568/Approche%20hybrid%20de%20description%20de%20proc%C3%A9d%C3%A9%20m%C3%A9tier.pdf?sequence=1&isAllowed=y>.

Annexe A

Service 1

A.1 classe User implémenté en java

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import java.io.Serializable;

@Entity
public class User implements Serializable {
    @Id
    @GeneratedValue
    private int id;
    private String name;
    public user() {
    }
    public user(String name) {
        this.name = name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

A.2 créer l'interface UserRepository

```
import org.springframework.data.jpa.repository.  
JpaRepository;  
  
public interface userRepository extends JpaRepository<user ,  
    Integer>{  
}
```

A.3 classe DummyDataCLR en java

```
import org.springframework.beans.factory.annotation.  
Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.stereotype.Component;  
  
import java.util.stream.Stream;  
  
@Component  
class DummyDataCLR implements CommandLineRunner {  
  
    @Override  
    public void run(String ... strings) throws Exception {  
        Stream.of("Pencil" , "Book" , "Eraser") . forEach(s->  
            userRepository . save(new user(s)));  
        userRepository . findAll() . forEach(s->System.out.println(s .  
            getName()));  
    }  
  
    @Autowired  
    private userRepository userRepository;  
}
```

Annexe B

ConfigService

B.1 classe ConfigServiceApplication en java

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.
SpringBootApplication;
import org.springframework.cloud.config.server.
EnableConfigServer;

@EnableConfigServer
@SpringBootApplication
public class ConfigServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(ConfigServiceApplicatin.class, args
    );
}
}
```

B.2 json

```
{
  name: "service-1",
  profiles: [
    "master"
  ],
  label: null,
  version: "6e1ea61d706133e2d8b62f40c6b784192fb58e8a",
  state: null,
  propertySources: [
    {
      "name": "file:/etc/config/service-1"
    }
  ]
}
```

```
name: "file:./src/main/resources/myConfig/application.properties",
source: {
  global: "xxxxx"
}
}
```

B.3 json

```
{
  name: "service-1",
  profiles: [
    "master"
  ],
  label: null,
  version: "6
e1ea61d706133e2d8b62f40c6b784192fb58e8a",
  state: null,
  propertySources: [
    {
      name: "file:./src/main/resources/
myConfig/user-service.properties",
      source: {
        me: "Djamel.Zerrouki@jimmi.
fr"
      }
    },
    {
      name: "file:./src/main/resources/
myConfig/application.properties",
      source: {
        global: "xxxxx"
      }
    }
  ]
}
```

B.4 classe userRestService en java

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.
RequestMapping;
```

```
import org.springframework.web.bind.annotation.  
RestController;  
  
 @RestController  
 public class userRestService {  
  
 @Value("${me}")  
 private String me;  
  
 @RequestMapping("/messages")  
 public String tellMe(){  
 System.out.println("c'est moi qui ai repondu!");  
 return me;  
}  
}
```

Annexe C

Microservice DiscoveryService

EurekaServer

```
package tn.insat.tpmicro.discoveryservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.
    SpringBootApplication;
import org.springframework.cloud.netflix.eureka.
    server.EnableEurekaServer;

@EnableEurekaServer
@SpringBootApplication
public class DiscoveryServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(
            DiscoveryServiceApplication.class
            , args);
    }
}
```

Annexe D

Microservice ProxyService

ProxyServiceApplication

```
package tn.insat.tpmicro.discoveryservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.*;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@EnableZuulProxy
@EnableDiscoveryClient
@SpringBootApplication
public class ProxyServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(ProxyServiceApplication.class,
            args);
    }
}
```

Organigramme de CNR

Annexe E

L'organigramme de la CNR

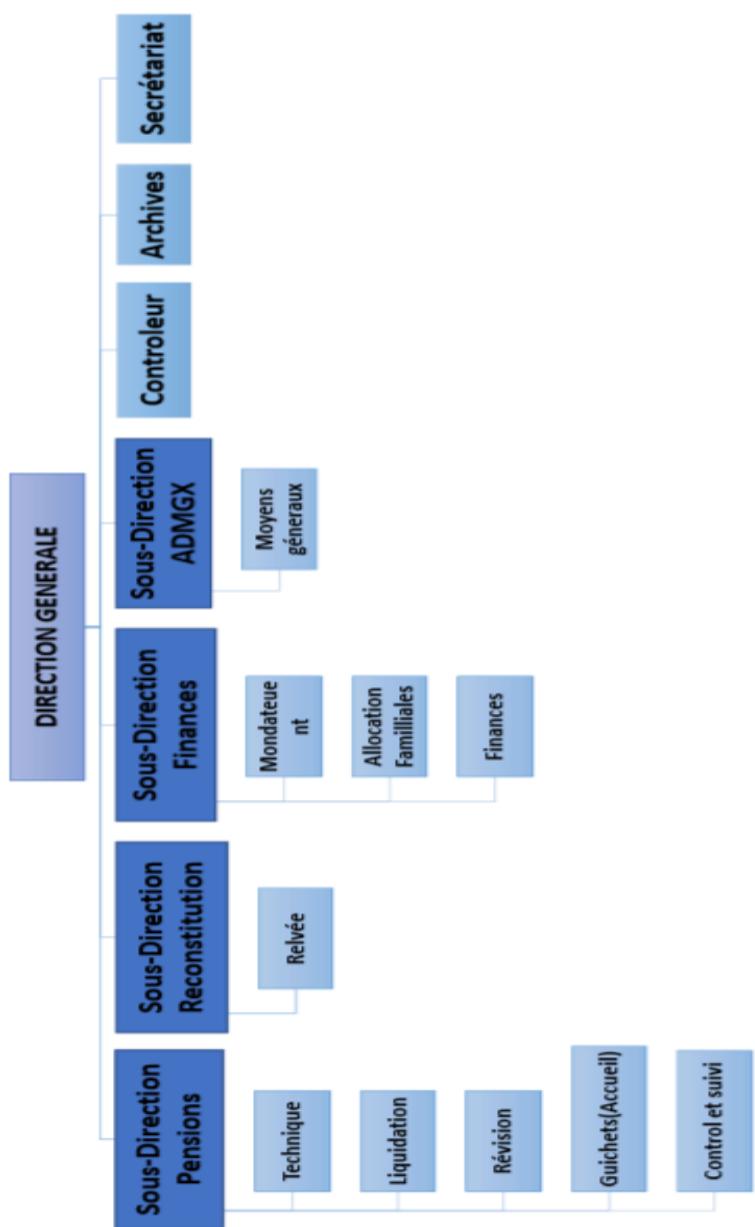


FIGURE E.1 – L'organigramme de la CNR