IMPLEMENTATION D'UNE APPLICATION INFORMATIQUE DISTRIBUEE POUR LA FACTURATION DES ABONNES DE LA REGIDESO / KISANGANI

Par
Vigeol MAVUNGU MAVUETA*
Assistant de deuxième mandat Matricule : 7977916L
Département : Informatique de Gestion Institut Supérieur Pédagogique d'Isangi
Téléphones : +243822570025, +243854447322

Corresponding Author:-

JUILLET 2022

SIGLES ET ABBREVIATIONS

- API : Applicative Programing Interface. L'interface de programmation applicative.
- CRUD : Create Read Update Delete.
- EJB : Enterprise JavaBeans
- JAX-RS : Java Api for RESTful Web Services.
- ORM : (Object-Relational Mapping) Mapping Objet relationnel.
- REST : Representational state transfer
- SQL: (Structured Query Language) Langage de requête structuré.
- UML: (Unified Modeling Language) Langage de Modélisation Unifié
- UP: (Unified Process) Processus Unifié.

RESUME

L'implémentation d'une application informatique distribuée constitue la tendance actuelle des équipes de développement logiciel des entreprises. Et pourtant, il se pose énormément des problèmes quant à sa réalisation compte tenu des multiples technologies qui entrent en compte dans sa mise en œuvre. Dans cet article, nous faisons le point sur les étapes indispensables dans la construction d'une application distribuée qui repose sur des architectures logicielles adaptées. C'est le cas de l'entreprise Regideso / Kisangani qui dans le besoin de maximiser ses recettes dans son processus de facturation de ses abonnés ne dispose pas d'un outil de collectes de données de consommation d'eau approprié et efficient auprès de ses abonnés. L'objectif de cet article est de définir les étapes analytiques et conceptuelle d'une application informatique distribuée reposant sur la technologie REST coté serveur consommée par une application mobile sous Android permettant à cette entreprise de disposer d'un système adapté à ses besoins et qui soit directement lié au reste de son infrastructure informatique.

ABSTRACT

The implementation of a distributed computer application is the current trend of software development teams in companies. However, there are a lot of problems in its realization due to the multiple technologies that are involved in its implementation. In this article, we review the essential steps in the construction of a distributed application based on adapted software architectures. This is the case of the company Regideso / Kisangani which, in the need to maximize its revenues in its billing process of its subscribers, does not have an appropriate and efficient tool for collecting water consumption data from its subscribers. The objective of this paper is to define the analytical and conceptual steps of a distributed computing application based on server-side REST technology consumed by a mobile application under Android allowing this company to have a system adapted to its needs and which is directly linked to the rest of its IT infrastructure.

I. INTRODUCTION

Le défi auquel sont confrontées les entreprises publiques de la république démocratique du Congo et plus particulièrement celles se trouvant dans la ville de Kisangani se caractérise par le manque d'un système d'information fiable et efficace qui soit en mesure de supporter les charges organisationnelles de ces entreprises. L'état actuel dans lequel se trouvent les entreprises tant publiques que privées de la ville de Kisangani soulève plusieurs problèmes liés à l'absence des moyens de traitements rapides et automatiques, et par-dessus de tout un système d'information adapté au besoin de ces entreprises.

Tel est le cas de la Regideso Kisangani, *« une entreprise à caractère technique jouissant de la responsabilité juridique régie par la disposition de la loi n°78-002 portant disposition applicable aux entreprises publiques »*, dans son processus de collectes des indicateurs de consommation de ses abonnés, de calcul et d'élaboration de factures de consommation d'eau, qui ne dispose pas d'un système automatisé pouvant faciliter ces différentes tâches qui jusqu'à l'heure de la rédaction de cet article recourt aux moyens traditionnels de collecte, nous comprenons par-là, l'utilisation des supports papiers. Cette tâche faisant partie du Système de traitement transactionnel comme l'ont défini K. Laundon et al. « Les Systèmes de traitement transactionnels sont la concrétisation des Systèmes Opérants. Ils exécutent et enregistrent les transactions quotidiennes et routinières associées aux évènements quotidien tels que la saisie des bons de commandes ou le calcul des tournées de livraison de la flotte de camions[1] », exige à cette entreprise de mettre en place une infrastructure logicielle et matérielle adéquate susceptible de prendre en charge la mobilité des agents qui doivent échanger les données ainsi prélevées avec le reste de l'infrastructure informatique de manière sure et valide.

Dans cette étude, la quintessence de notre problème de recherche se fonde sur la lenteur et les difficultés rencontrées dans la collecte des données de consommation d'eau auprès des abonnés, dans le calcul et l'établissement et la distribution des factures. Cette situation ne permettant pas aux abonnés de connaitre dans un délai acceptable le montant de leur consommation à payer. Cela occasionne un faible de taux de paiement des factures car ces dernières arrivent souvent lorsque la plupart de ménages ont déjà dépensé la grande partie de leur salaire. pour améliorer le processus de collecte de données de consommation d'eau auprès des abonnés de la Regideso ?

Cette étude recentre les besoins technologiques devant être implémentés dans le processus de facturation de consommation d'eau dans cette entreprise. L'hypothèse à émettre se fonde sur l'ajout d'une couche logicielle à l'infrastructure informatique déjà existante. Sur ce, la solution pouvant prendre en compte la mobilité des agents au terrain pour intégration solide avec le reste de l'infrastructure logicielle serait l'implémentation d'une nouvelle couche logicielle mobile sous Android reposant sur la technologie REST[2] facilitant les échanges entre les agents et le serveur d'application qui est hébergé au sein de l'entreprise. Cette solution permettrait à la Regideso / Kisangani d'améliorer les opérations de prélèvement des indicateurs des compteurs auprès de ses abonnés et de traitement automatiquement des données issues des abonnés en vue de déclencher les opérations de facturation dans un délai jugé satisfaisant.

Le principal objectif de cette étude est d'expliquer les différentes phases du développement d'une application informatique distribuée basée sur une architecture en 3 couches dont chacune des couches correspond à une brique logicielle bien identifiée Pour atteindre cet objectif, nous faisons recourt à la méthode UP (Unified Process) « Processus Unifié » en français, qui est un processus de développement logiciel construit sur UML ; il est itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques[3]. UP utilise UML pour la modélisation du système. UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et à décrire des besoins, spécifier, concevoir des solutions et communiquer des points de vue[4].

Pour estimer le cout du développement de cette application, nous avions utilisé la méthode COCOMO[5] (acronyme de l'anglais Constructive COst MOdel) qui est un modèle permettant de définir une estimation de l'effort à fournir dans un développement logiciel et la durée que ce dernier prendra en fonction des ressources allouées.

Les techniques d'observation et documentaires ont été utilisées pour nous permettre de nous imprégner de la situation réelle qui prévalait au sein de cette entreprise. Une attention particulière a ainsi été faite sur l'architecture logicielle qui ne tient plus compte de la grandeur de l'entreprise. Pour bien cerner le problème, nous avions subdivisé cette étude en 2 sections

---

Ainsi nous avons soulevé les préoccupations suivantes en adéquation avec le constat qui a été fait, qui de fait, constituent la quintessence de notre problématique : Quelle solution adoptée

1 Kenneth L., Jane L., Eric F., Serge C. et Sophie C., management des systèmes d'information, 13ième édition, Nouveaux Horizons, 2013, page 51.

2 Antonio Goncalves, Java EE 6 et GlassFish 3, Pearson Education France, 2010, ISBN : 978-2-7440-4157-0, p.

3 Pascal Roques et Franck Vallée, UML 2 en action, De l'analyse des besoins à la conception 4e édition, Eyrolles 2007, p.12.

4 Pitman, N., *UML 2 en concentré.* O'Reilly 2006, p. 15

5 https://fr.wikipedia.org/wiki/Constructive_Cost_Model consulté le 11 Mars 2022 à 13h42

dont la première se focalise sur les approches théoriques, explicitant au clair les notions essentielles des architectures logicielles existantes et les tendances technologiques actuelles. La seconde section s'intéresse premièrement à l'analyse fonctionnelle et organique de l'actuel système allant de la collecte de données jusqu'à la facturation des abonnés. En second lieu, se présente la conception de la nouvelle architecture logicielle en justifiant les raisons sur lesquelles s'est fondée la décision.

II. CADRE THEORIQUE

II.1. DEFINITION DES CONCEPTS

- **Abonné** : C'est le souscripteur, personne physique ou morale d'une police d'abonnement et fourniture d'eau[6].
- **Facturation** : La facturation à la REGIDESO Kisangani tient compte de deux éléments à savoir : Le Mètre cube facturé (M^3) et le Tarif par catégorie de l'abonné.
  - Le Mètre Cube facturé : C'est la consommation de l'abonné. Un abonné disposant d'un compteur, ces indexes du mois sont comparés aux indexes du mois passé et la différence qui se dégage.
  - Le Tarif : Il est subdivisé en 7 catégories tarifaires : Les Bornes fontaines, les Domestiques, les Intermédiaires, les Commerciales, les Industrielles, les Instances Officiel, les Domestiques Spéciaux.

- **Architecture**

L'architecture spécifie la structure d'un système. On parle d'architecture fonctionnelle pour définir les services du système, d'architecture technique pour les composants techniques utilisés et d'architecture applicative pour décrire le découpage en sous-systèmes[7].

Pour mieux cerner le concept d'architecture qui est souvent mal compris parce qu'on la situe dans la structure résultante d'un modèle. L'architecture n'est pas la conséquence du modèle mais préside au contraire à son organisation. Cette organisation fixe des directives générales au développement, et les structures qu'elle induit aident au maintien de l'intégrité du modèle[8]. Ainsi en empruntant les termes de Pascal Roque et Franck Vallée, l'architecture est définie comme : *l'ensemble des décisions d'organisation du système logiciel qui défend les intérêts de son propriétaire final. Les intérêts s'expriment en termes d'exigences fonctionnelles, techniques et économiques. L'architecture y répond par l'intégration de plusieurs styles de développement informatique qu'elle adapte aux éléments logiciels d'un contexte existant*[9].

Nous passons en suite en revue différentes architectures logicielles existantes qui constituent le socle de toutes les applications informatiques.

---

6 Statut de la regideso, p.15.

7 Goncalves A., les Cahiers du Programmeur : Java EE 5, éditions Eyrolles, 2007, ISBN : 978-2-212-12038-7, p. 42.

8 Pascal Roques et Franck Vallée, op. cit, p.37

II.2. ARCHITECTURES EXISTANTES

Plusieurs architectures existent dans le monde de développement de logiciels. Nous les présentons comme suit :

- **Architecture simple tiers ou architecture monolithique**

Une architecture monolithique est une architecture constituée d'un seul bloc et s'exécutant sur une seule machine. Ces applications sont généralement utilisées dans le domaine du temps réel ou bien au sein d'applications demandant de grandes performances. Elles restent également omniprésentes dans le monde du grand public, étant utilisées en standalone (de manière autonome) sur les machines personnelles[10].

Cette architecture monolithique est appelée simple tiers car toutes les fonctionnalités sont comprises dans une seule couche logicielle[11]. Cette architecture a les inconvénients suivants : évolution difficile, maintenance lourde, partage difficile des données, etc. Compte tenu de ses inconvénients, les architectes de systèmes logiciels ont pensé à d'autres solutions stimulées par l'essor des réseaux et surtout de l'internet.

- **Architecture Client Serveur**

L'organisation du client/serveur de données est la suivante : diverses stations de travail, les clients peuvent dialoguer avec un serveur de données sur lequel sont installés le moteur de la base de données et les données à gérer. Les postes clients ne possèdent donc pas la base de données mais transmettent au serveur les données des requêtes (le plus fréquemment en utilisant le langage SQL). Ce type d'application offre à l'utilisateur une interface riche, tout en garantissant la cohérence de données qui restent gérées de façon centralisée. L'architecture à 2 niveaux ne peut fonctionner que grâce à une couche logicielle particulière, logiciel médiateur ou middleware qui assure la liaison transparente entre le client et le serveur à travers un réseau : transport des requêtes, harmonisation des types de données, respect des protocoles de gestion de la performance[12].

La raison de cette approche est de centraliser au maximum les données afin de permettre à plusieurs utilisateurs d'y accéder simultanément. Les données peuvent ainsi être partagées entre plusieurs utilisateurs de l'application. Cette architecture est communément appelée client- serveur, qui dans notre approche peut être représentée en deux tiers[13]. La couche « client » est souvent de type « léger » c'est-à-dire utilisant un navigateur web. Certains langages de programmation tels que « Java » et « C# » offrent la possibilité d'utiliser les interfaces graphiques faisant appels aux ressources du système d'exploitation pour les construire, on les appelle « client lourd ». La couche « serveur » quant à elle, se charge de fournir les ressources demandées par le client. Un des inconvénients de cette architecture est que la logique chargée de la manipulation des données et de l'application des règles métiers afférentes est incluse dans l'application elle-même. Cela pose problème lorsque plusieurs applications doivent partager l'accès à une base de données[14].

- **Architecture trois-tiers**

Ce modèle est une évolution du modèle d'application client-serveur défini précédemment. L'architecture trois-tiers est donc divisée en trois niveaux ou rangées : Le tiers client qui représente la machine sur laquelle l'application cliente est exécutée. Le tiers métier qui représente la machine sur laquelle l'application centrale est exécutée. Le tiers accès aux données qui correspond à la machine gérant le stockage des données[15]. La séparation entre le client, l'application et le stockage, est le principal atout de ce modèle. Toutefois, dans des architectures qui demandent de nombreuses ressources, il sera assez limité. En effet, aucune séparation n'est faite au sein même de l'application, qui gère aussi bien la logique métier que la logique fonctionnelle ainsi que l'accès aux données.

9 Pascal Roques et Franck Vallée, op. cit, p.38

10 Frédéric Chuong, Olivier Corgeron Cyril Joui, Jean-Baptiste Renaux et Maxime Vialette, op. cit., p.2

11 Gérard CASANOVA - Denis ABÉCASSIS, Gestion de projet - calculs des coûts, Université de Lorraine, 2014, p.11

- **Architecture Multi-tiers**

Dans le cadre d'applications beaucoup plus importantes, l'architecture trois-tiers montre ses limites. L'architecture multi-tiers est simplement une généralisation du modèle precedent qui prend en compte l'évolutivité du système et évite les inconvénients de l'architecture trois-tiers vus précédemment. Dans la pratique, on travaille généralement avec un tiers permettant de regrouper la logique métier de l'entreprise. L'avantage de ce système, c'est que ce tiers peut être appelé par différentes applications clientes, et même par des applications classiques, de type fenêtrées, qui ne passent donc pas par le serveur Web.

II.3.CHOIX DE L'ARCHITECTURE A UTILISER

Le but de cet article est d'expliciter les étapes essentielles pour la mise en place d'une application informatique distribuée. Comme nous l'avions évoqué au début de cette rédaction, l'application que nous envisageons de développer se fonde sur l'architecture à trois-tiers et sur ce, nous l'implémentons une API REST en Java qui va être consommée par un client Android. Ainsi nous allons dans la section qui va suivre, fixer le cadre analytique et conceptuel de ce choix.

III. ANALYSE ET CONCEPTION

Dans cette section, nous évoquons l'analyse et la conception à faire pour bâtir un modèle pouvant nous servir de socle pour la suite de notre étude.

III.1. ANALYSE

Cette analyse se caractérise sur la capture des besoins fonctionnels et celle des besoins techniques. Les besoins fonctionnels ont pour objectif de décrire précisément l'ensemble des fonctions d'un logiciel ou d'une application, et de fixer ainsi le périmètre fonctionnel du projet[16]. C'est dans ce document que nous précisons l'étude du contexte fonctionnel du système, en décrivant les différentes façons qu'auront les acteurs d'utiliser le futur système. Tandis que la capture des besoins technique couvre, par complémentarité avec celle des besoins fonctionnels, toutes les contraintes qui ne traitent ni de la description du métier des utilisateurs, ni de la description applicative[17].
Pour concrétiser notre démarche, nous présentons les besoins fonctionnels et techniques de cette étude.

III.1.1. CAPTURE DE BESOINS FONCTIONNELS

L'application que nous envisageons de réaliser pour la Regideso/Kisangani permettra de :

- Prélever les indicateurs du compteur de robinet de consommation d'eau de l'abonné ;
- Calculer rapidement le volume d'eau consommé par l'abonné;
- Lancer automatiquement les impressions des factures juste après prélèvement des indicateurs ;
- Gérer les utilisateurs du système ;
- Gérer les abonnés ;
- Gérer les agents de terrain ;

Les acteurs ainsi recensés dans cette étude sont :

- Abonné ;
- L'agent du terrain (Préleveur) ;
- Le chef de service des opérations;
- Le chef de la facturation.

---

12 M.Bigaud, JP Bourey, H. Camus, D. Corbeel, conception des systèmes d'information : Modélisation de données, étude des cas, Edition TECHNIP, Paris, 2006, page 14.

13 Robert ENGLANDER, Java et SOAP, O'Reilly, 2009, page 45.

14 http://remy-manu.developpez.com/introjava2ee consulté le 28 février 2022

15 Frédéric Chuong, Olivier Corgeron Cyril Joui, Jean-Baptiste Renaux et Maxime Vialette, op. cit, p.3

16 https://www.lecoindesjeux.com/rediger-une-specification-fonctionnelle-detaillee/ consulté le 14 avril 2022

17 Pascal Roques et Franck Vallée, op. cit, p.93

Cette phase représente un point de vue « fonctionnel » de l'architecture système. Par le biais des cas d'utilisation, nous serons en contact permanent avec les acteurs du système en vue de définir les limites de celui-ci, et ainsi éviter de trop s'éloigner des besoins réels de l'utilisateur final. Les cas d'utilisation constituent un moyen de recueillir et de décrire les besoins des acteurs du système. Ils peuvent être aussi utilisés ensuite comme moyen d'organisation du développement du logiciel, notamment pour la structuration et le déroulement des tests du logiciel[18].

Diagramme de cas d'utilisation de notre étude

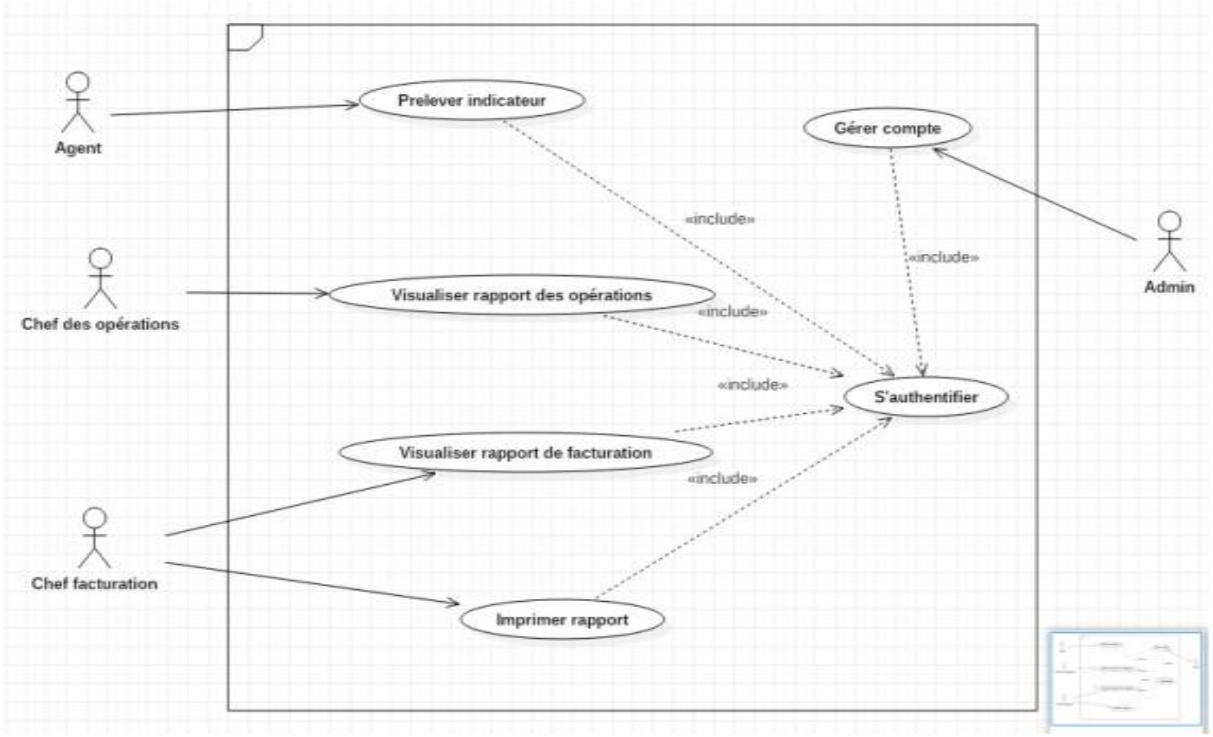

Figure I : Diagramme de cas d'utilisation

Description textuelle de diagramme de cas d'utilisation

- Cas d'utilisation Prélever indicateur
  - Résumé : ce cas d'utilisation permet de prélever les indicateurs au compteur de l'abonné.
  - Acteurs : Agent.
  - Dates : le 11 juin 2022
  - Responsable : Mavungu.
  - Version : 1.

18 Joseph Gabay et David Gabay, UML 2. Mise en œuvre guidée avec études de cas : ANALYSE ET CONCEPTION, Dunod, Paris, 2008, p. 61

- Cas d'utilisation Gérer compte
  - Résumé : Ce cas permet d'enregistrer, modifier, afficher et supprimer les comptes utilisateurs marchandises dans la base de données.
  - Acteurs : Administrateur.
  - Dates : le 11 juin 2022.
  - Responsable : Mavungu
  - Version : 1.0
- Cas d'utilisation Visualiser rapport des opérations
  - Résumé : ce cas d'utilisation permet de visualiser le rapport des opérations de prélèvements envoyées par les agents sur terrain.
  - Acteurs : Chef des opérations
  - Dates : le 11 juin 2022
  - Responsable : Mavungu
  - Version : 1.
- Cas d'utilisation Authentification
  - Résumé : ce cas permet de vérifier l'authenticité de l'utilisateur.
  - Acteurs : Agent, Administrateur, Chef des opérations, chef de facturation.
  - Dates : le 11 juin 2022
  - Responsable : Mavungu
  - Version : 1
- Cas d'utilisation Visualiser les opérations de Facturation
  - Résumé : ce cas permet de voir sur un tableau toutes les facturations effectuées par le système en fonction des données recueillies auprès des abonnés.
  - Acteurs : Chef de facturation
  - Dates : le 11 juin 2022
  - Responsable : Mavungu
  - Version : 1
- Cas d'utilisation Imprimer Factures
  - Résumé : ce cas permet de lancer les impressions.
  - Acteurs : Chef de facturation
  - Dates : le 11 juin 2022
  - Responsable : Mavungu
  - Version : 1

III.1.2. CAPTURE DES BESOINS TECHNIQUES

La capture des besoins techniques pour cette étude, représente le volet technique de notre application. Elle est constituée des couches logicielles qui sont reparties en 5. Une couche logicielle représente un ensemble de spécifications ou de réalisations qui respectivement expriment ou mettent en œuvre un ensemble de responsabilités techniques et homogènes pour un système logiciel[19].

Les couches logicielles qui constituent notre application sont détaillées avec des technologies spécifiques comme suit :

- Couche présentation : cette couche restitue les données à l'utilisateur et transforme les actions en évènements de l'application. Dans notre cas cette couche sera implémentée par une application Android permettant à l'agent de saisir les indicateurs de consommation d'eau auprès des abonnés.
- Couche Application : Elle représente les objets de contrôle et pilote les règles de l'application, y compris les règles d'échanges entre application. Cette couche dans notre étude est implémentée par un service web de type REST qui publie les objets en utilisant le format XML et JSON permettant ainsi un échange entre la couche présentation et la couche métier.
- Couche Métier : elle représente les objets du métier et implémente leurs règles de gestion. Cette couche est matérialisée dans notre étude par un contrôleur EJB qui exécute le code métier de l'application.
- Couche Accès aux données : elle restitue les représentations métiers à partir des moyens de stockage. Elle est implémentée dans notre étude par les classes d'entités en Java. Elle fait recours à la technologie ORM.
- Couche stockage des données : elle assure la persistance des stockages. Dans notre étude, cette couche est implémentée par la solution JPA.

Les cinq couches constituent les briques logicielles de notre application. Son déploiement sera fait à partir d'un serveur d'application Glassfish 5.

III.1.3. ESTIMATION DE CALCUL DU COUT DE DEVELOPPEMENT LOGICIEL

L'estimation est une activité complexe qui implique différentes pratiques, de nombreux acteurs (client, directeur, chef de projet, développeurs, etc.) et un environnement dynamique et

19 Pascal Roques et Franck Vallée, op. cit., p.104

incertain (dû aux changements des exigences du client, à l'évolution des compétences des développeurs, à la disponibilité des ressources, etc.). Ces éléments sont des facteurs d'échec principaux de l'estimation de l'effort, coûts et durée de projets logiciels[20] .

Le calcul des coûts consiste en un exercice exigeant et délicat qui sera affiné pendant toute la phase préparatoire du projet. La principale source de difficultés est liée à l'estimation d'un produit nouveau, encore mal défini et qu'il faudra pourtant chiffrer. Le maître d'œuvre doit connaître le coût du projet avant d'être trop engagé dans sa réalisation, de manière à pouvoir réorienter ses choix, ou renoncer à son projet[21].

Comme énoncé à l'introduction de cet article, nous avons recours à la méthode d'estimation de cout COCOMO qui s'appuie uniquement sur la taille estimée du logiciel et sur le type de logiciel à développer. Nous allons estimer également le cout des matériels associés à cette application. Ainsi, pour le faire, nous imaginons l'architecture matérielle de notre application pour déterminer les équipements qui seront utilisés.

a. Architecture du système proposé

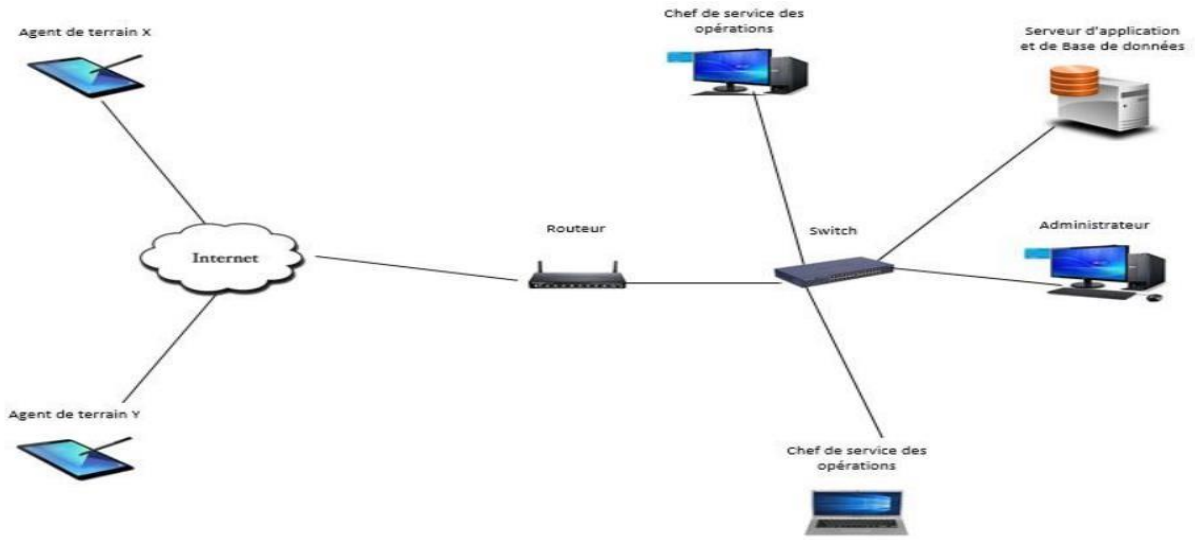

Figure 11 : Architecture du système.

Ce scénario regroupe les terminaux mobiles d'agent de terrain, le poste du chef de service des opérations, le poste du chef de service de facturation, l'administrateur principal de l'application, le serveur d'application et de base de données.

[20] Safae Laqrichi. Approche pour la construction de modèles d'estimation réaliste de l'effort/coût de projet dans un environnement incertain : application au domaine du développement logiciel, thèse de doctorat, Université de Toulouse, Informatique, délivré par l'Ecole des Mines d'Albi-Carmaux, 2015, page 13

[21] Gérard CASANOVA - Denis ABÉCASSIS, op. cit., page 11

b. Besoins matériels et logiciels

Les besoins matériels et logiciels de ce scénario sont présentés dans le tableau suivant :

Catégorie	Désignation	Quantité	Prix unitaire FC	Prix total FC	Commentaire
Matériels et logiciels	Ordinateur de bureau	2	1000000	2000000	A acheter
	Ordinateur portable	1	800000	800000	A acheter
	Switch	1	100000	100000	A acheter
	Serveur (Application et Base de données)	1	5125000	5125000	A acheter
	Tablette Android	2	184500	369000	A acheter
	Câble UTP	1 rouleau (300m)	202000	202000	A acheter
	GlassFish Server 5.0	[redacted]			Gratuit
	NetBeans 8.2				Gratuit
	JDK 8				Gratuit
	MySQL Serveur 5				Gratuit
	Java SE et Java EE				Gratuit
	Cout total de matériels et logiciels			**8596000**	

Tableau 1 : Besoins matériels et logiciels.

c.Cout de développement et formation des utilisateurs du scénario

Un projet de type mode semi-détaché convient au mieux pour ce scénario. On estime le nombre de lignes de notre code source à 8KLoc qui représente approximativement 8000 de lignes de codes. Nous servant de la formule fournie par COCOMO nous pouvons ainsi estimer le cout du développement de l'application et le cout global.
Cout de développement :

Intitulé	Formule	Valeurs
Effort à consentir	Effort = $3.0*(8)^{1.12}$	30HM
Temps de développement	TDev=$2.5*(30)^{0.35}$	8.22 Mois
Nombre moyen des personnes	Effort / TDev	4 personnes
Cout financier en FC	Effort * Salaire moyen congolais[22]	**5460000 FC**

Tableau 2 : Cout de développement logiciel.

[22] Salaire moyen = SMIG * nbJours. (DECRET MIN. N°18/017 DU 22 MAI 2018, page 5). Dans notre cas le salaire moyen est égal à 188000Fc.

Cout global :

Désignation	**Montant en Francs Congolais**
Cout total des matériels et logiciels	8596000
Cout de développement	5460000
Cout de formation des utilisateurs	400000
Cout total du scénario	**14456000**

Tableau 3 : Cout de total de l'implémentation.

III.2. CONCEPTION Diagramme de classes

Le diagramme de classes a toujours été le diagramme le plus important dans toutes les méthodes orientées objet. C'est également celui qui contient la plus grande gamme de notations et de variantes. UML a réussi à unifier le vocabulaire et les concepts sans perdre la richesse et les apports des différentes méthodes existantes[23].

Le diagramme de classe constitue l'un des pivots essentiels de la modélisation avec UML. En effet, ce diagramme permet de donner la représentation statique du système à développer. Cette représentation est centrée sur les concepts de classe et d'association. Chaque classe se décrit par les données et les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes[24].

Les règles de gestion :

- Un agent peut prélever un ou plusieurs indicateurs de consommation ;
- Un indicateur concerne un et un seul abonné ;
- Un compte appartient à un et un seul agent ;
- Un compte appartient à un chef de service ;
- Un compte appartient à un chef de facturation ;

23 Pascal Roques et Franck Vallée, op. cit., p.133

24 Joseph Gabay et David Gabay, op. cit., p.17

Le diagramme de classes correspondant à notre étude se présente comme suit :

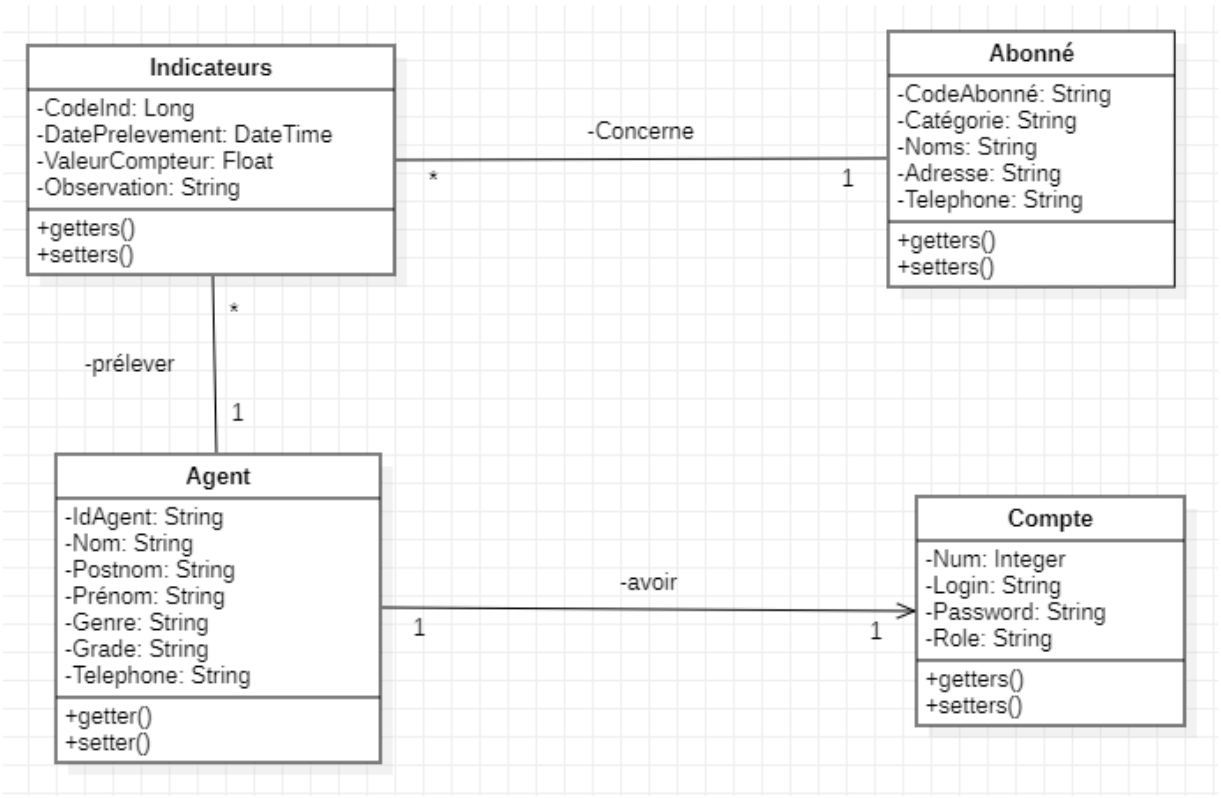

Figure 2 : Diagramme de classes

Diagramme de séquences

Le diagramme de séquence décrit les interactions entre un groupe d'objets en montrant, de façon séquentielle, les envois de message qui interviennent entre les objets. Le diagramme peut également montrer les flux de données échangées lors des envois de message. Pour interagir entre eux, les objets s'envoient des messages. Lors de la réception d'un message, un objet devient actif et exécute la méthode de même nom. Un envoi de message est donc un appel de méthode[25].

Ce diagramme s'appuie sur les cas d'utilisation élaborés dans le diagramme de cas d'utilisation. Autant qu'il y a des cas d'utilisation ainsi ce sera le nombre des diagrammes de séquences dans cette partie. Nous avions énuméré six cas d'utilisation et par conséquent nous aurons le même nombre des diagrammes de séquences.

[25] Fien VAN DER HEYDE et Laurent DEBRAUWER, UML 2 Initiation, exemples et exercices corrigés 2ième édition, Editions ENI, p.46

- Diagramme de séquence gérer compte

Dans ce diagramme de séquence nous avions réparti les opérations en 4 taches : Créer compte, modifier compte, supprimer compte et afficher compte.
Diagramme de séquence Création Compte Utilisateur.

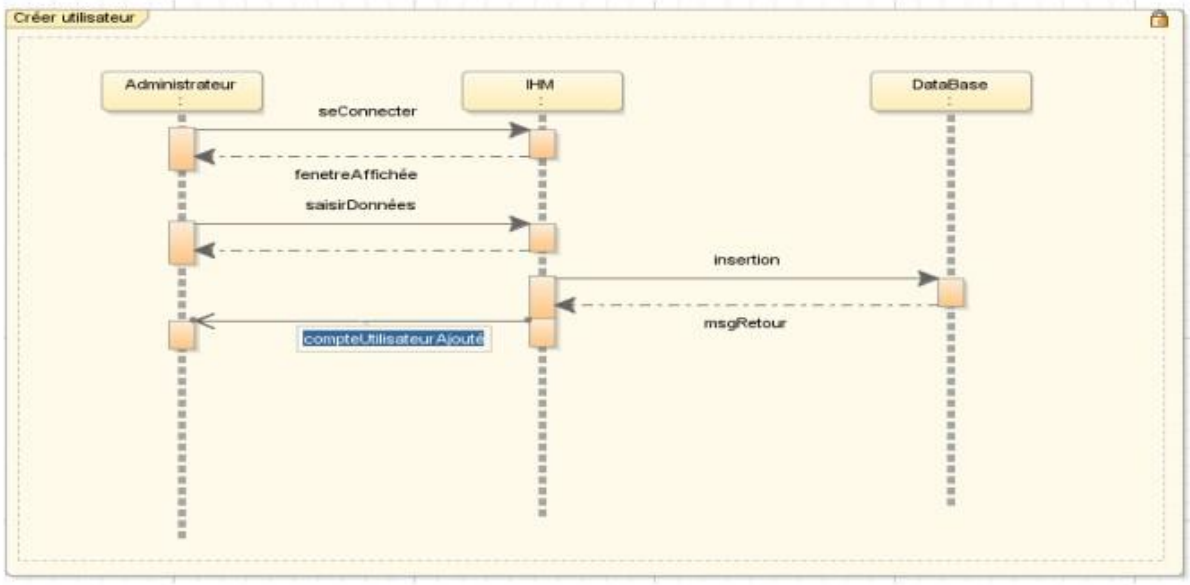

Figure 3 : Diagramme de séquence création compte utilisateur

Diagramme de séquence Modification compte utilisateur.

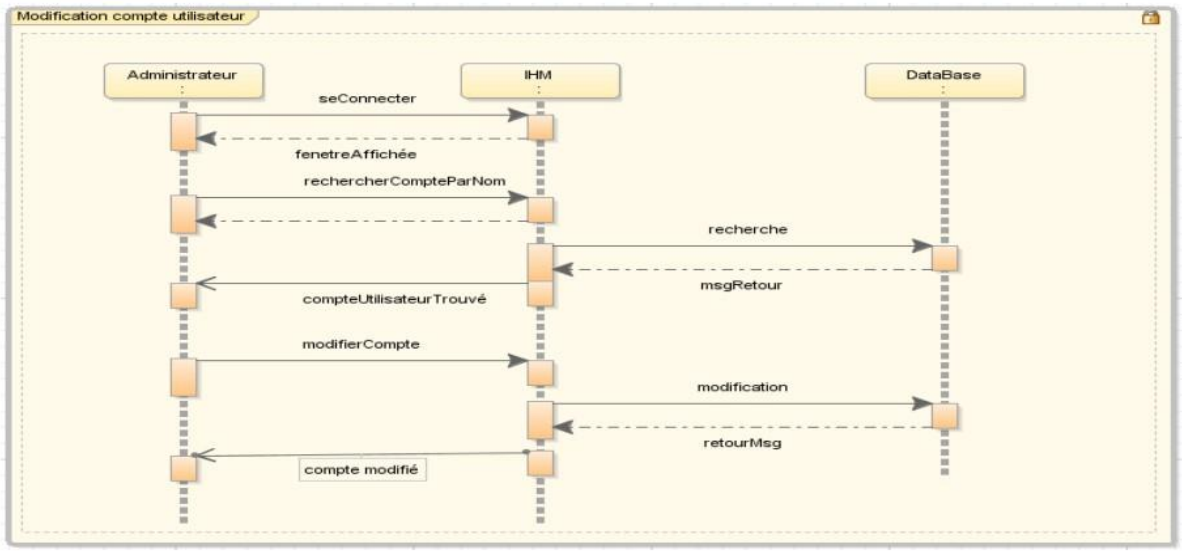

Figure 4 : Diagramme de séquence modification compte utilisateur

Diagramme de séquence Rechercher compte utilisateur

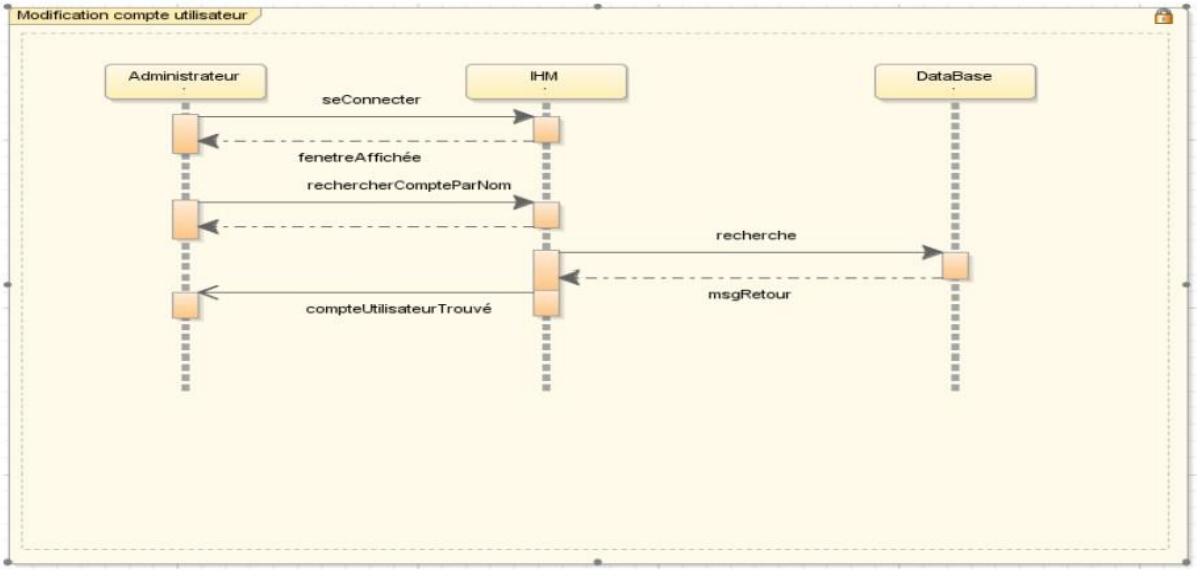

Figure 5 : Diagramme de séquence Rechercher compte utilisateur

Diagramme de séquence Suppression compte utilisateur

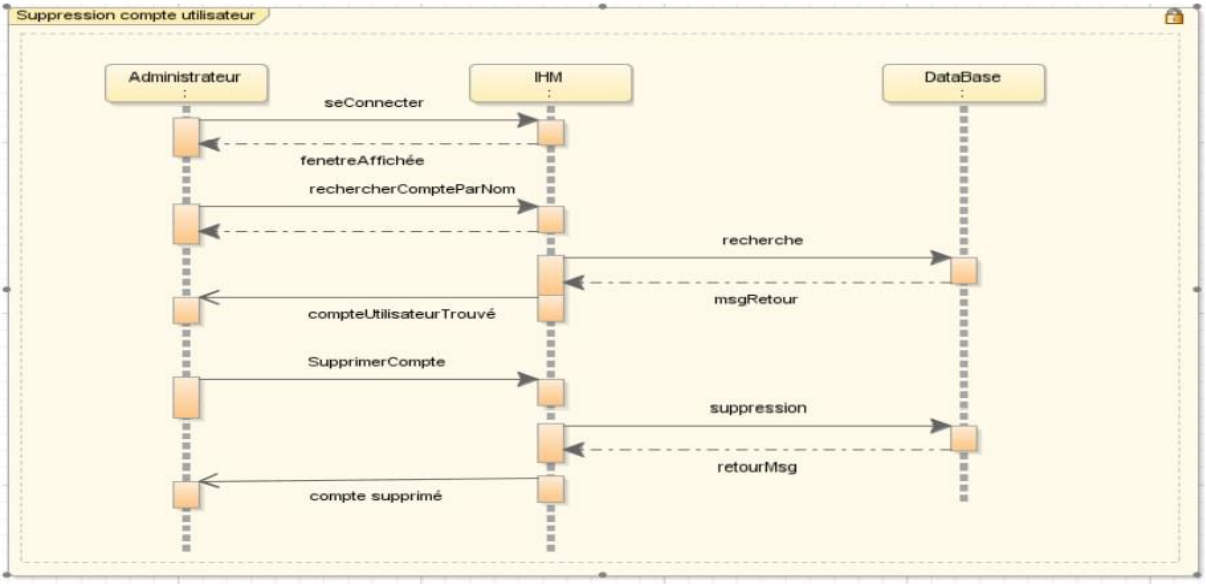

Figure 6 : Diagramme de séquence Suppression compte utilisateur

- **Diagramme de séquence s'authentifier**

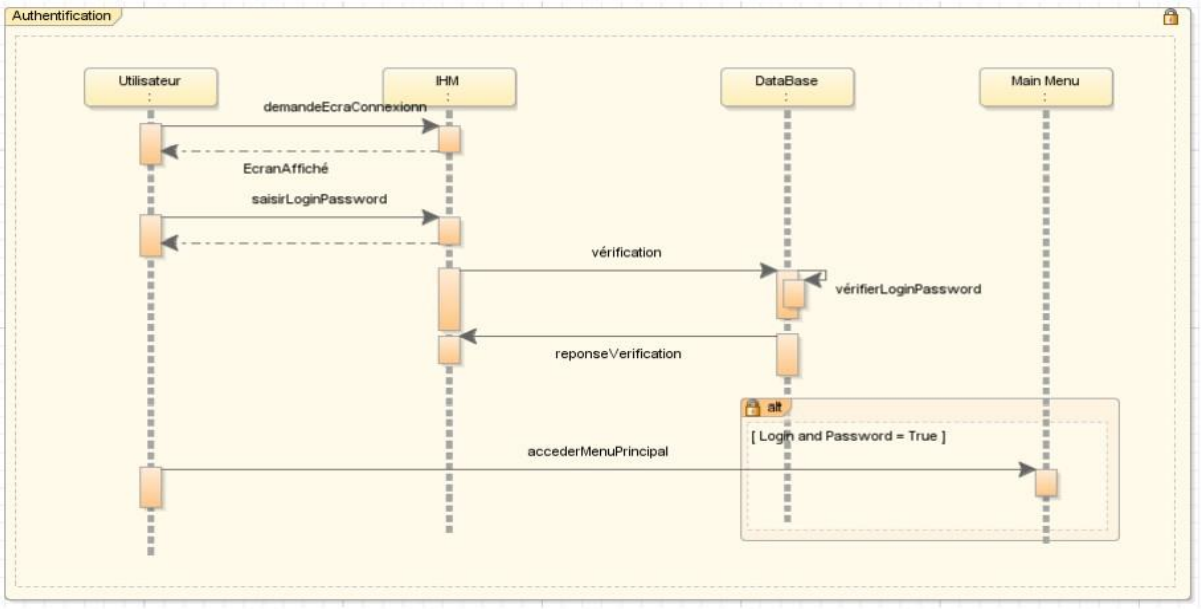

Figure 7 : Diagramme de séquence S'authentifier

- **Diagramme de séquence « Visualiser rapport des opérations »**

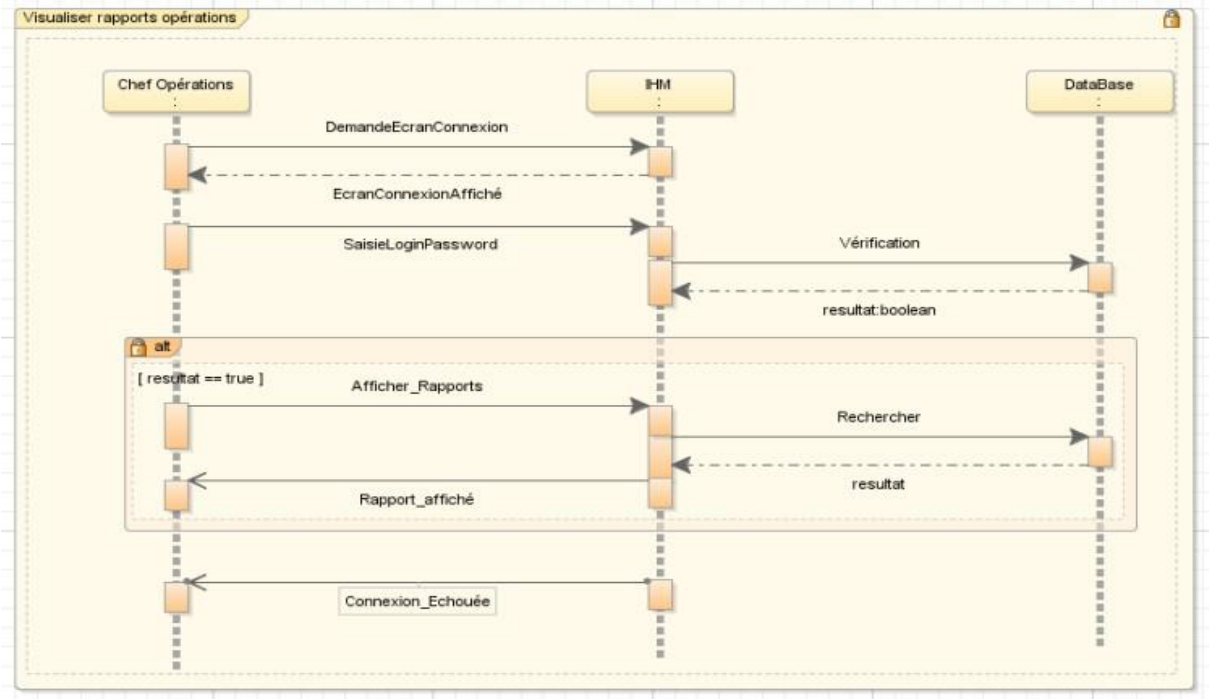

Figure 8 : Diagramme de séquence « Visualiser rapport des opérations »

Diagramme de séquence « Visualiser rapport des facturations »

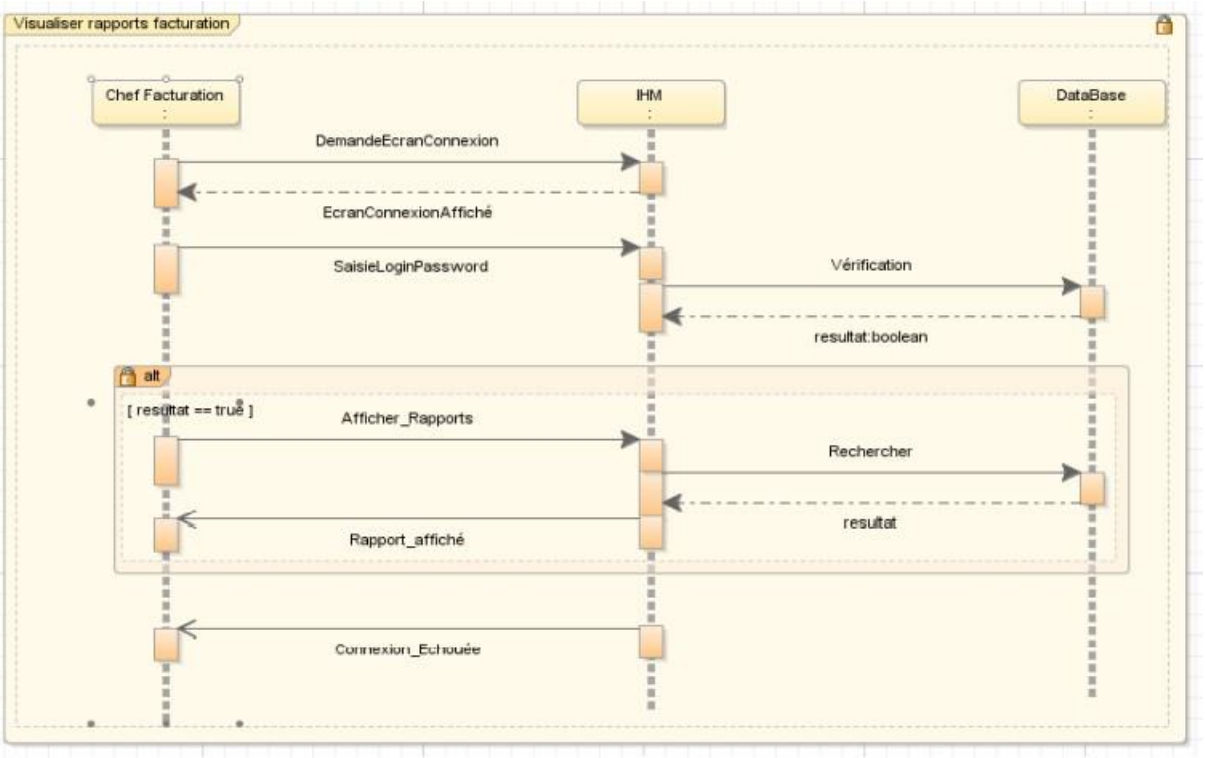

Figure 9 : Diagramme de séquence « Visualiser rapport des facturations »

Diagramme de séquence « Prélever indicateurs »

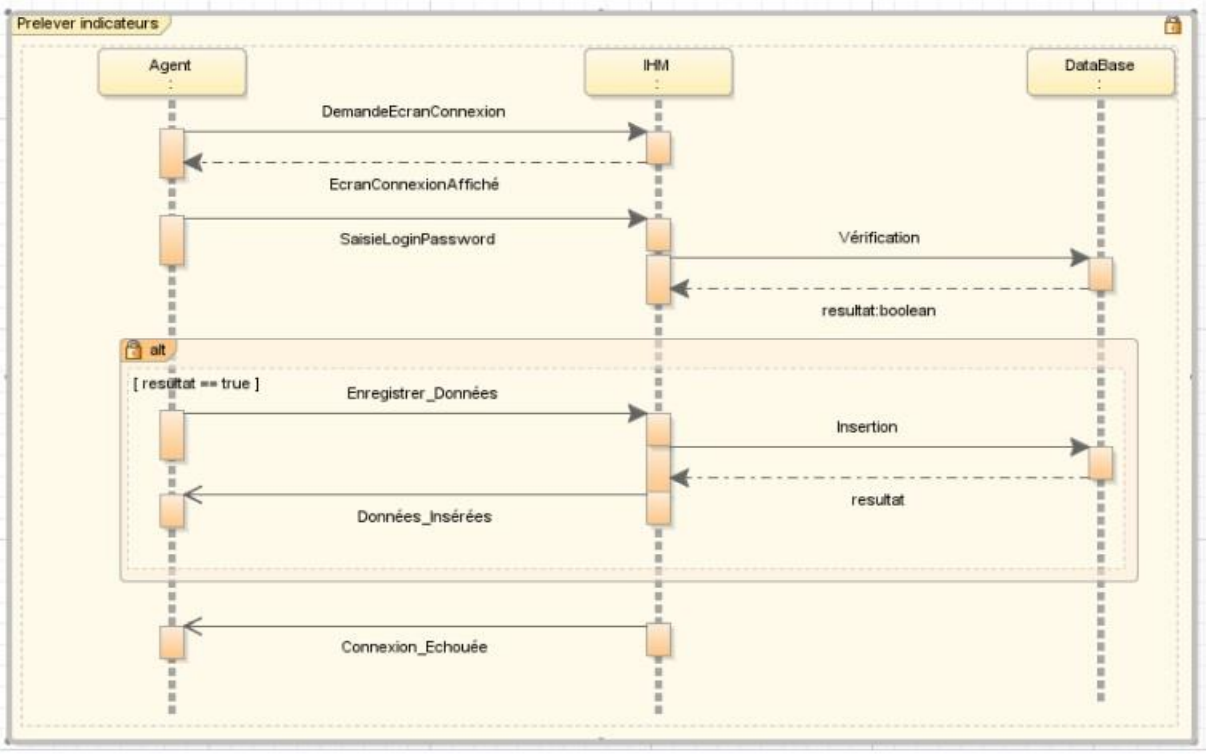

Figure 10 : Diagramme de séquence « Prélever indicateurs »

- **Diagramme de séquence « Imprimer factures »**

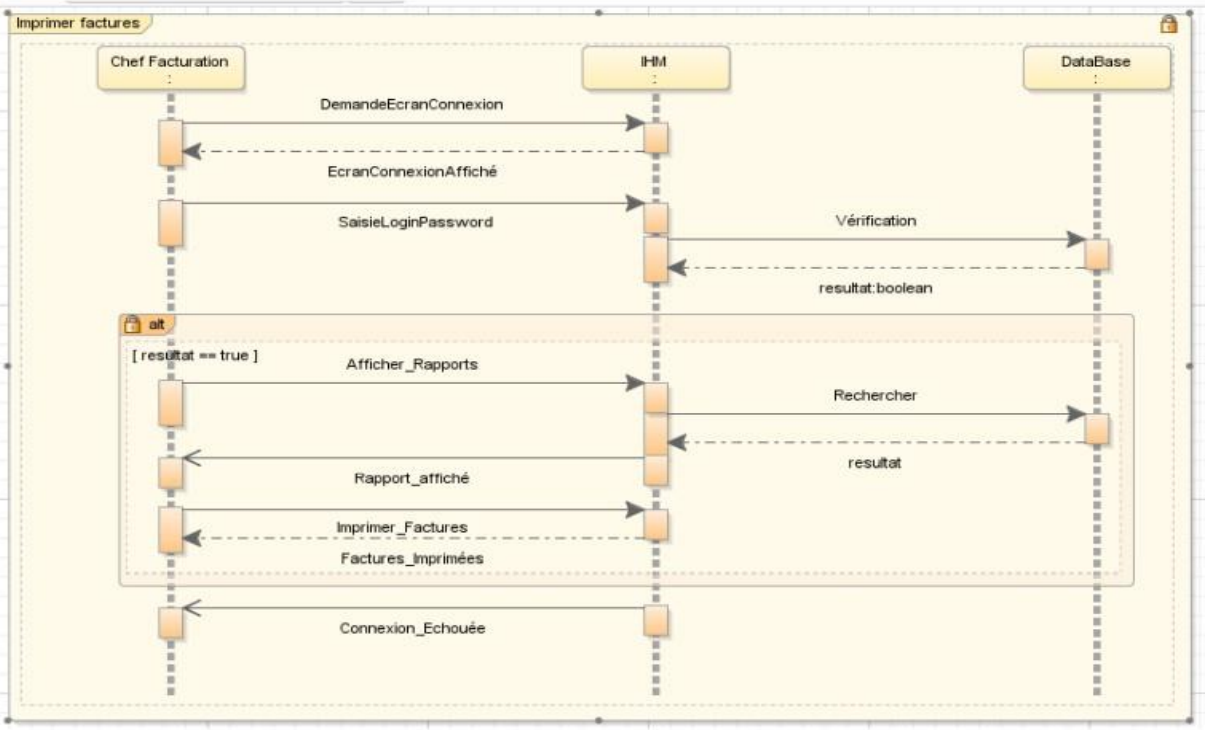

Figure 11 : Diagramme de séquence « Imprimer factures »

IV. PRESENTATION DES TECHNOLOGIES

Dans cette dernière section, nous présentons notre application globale qui sera constituée de deux parties : L'application Back-End et Front-End. Diverses technologies seront utilisées pour la réalisation de cette application.

IV.1. APPLICATION BACK-END

Le Back-End, c'est la partie immergée de l'iceberg. Elle est invisible pour les utilisateurs finaux mais représente une grande partie du développement d'une application distribuée. Sans elle, l'application reste une coquille vide. Le développement de notre application coté serveur s'appuie sur la plateforme Java EE. Nous déploierons un service web de type REST disposant des opérations CRUD dans un serveur d'application Glassfish 5. En Java EE l'implémentation de référence pour développer un service web est JAX-RS.

IV.2. APPLICATION FRONT-END

Une application Front-End ou frontale peut être l'interface directe avec les utilisateurs et envoyer les demandes à un programme distant en arrière-plan situé sur un autre ordinateur pour obtenir les données requises ou exécuter le service demandé. Dans un modèle informatique client/serveur, le « front-end » est généralement un client et le « back-end », un serveur.

Dans cette étude, nous avons fait le choix de la technologie Android qui va constituer le client à implémenter. Pour réaliser cette application Android nous avons recours à l'EDI Android Studio 4.

CONCLUSION

Nous voici à la fin de notre article intitulé mise en place d'une application distribuée pour la gestion des facturations d'eau au sein de la Regideso Kisangani. Il était question dans cette étude d'implémenter une solution informatique capable de répondre efficacement aux difficultés que connaissent les agents qui prélèvent les indices de compteurs de consommation d'eau auprès des abonnés.

Compte tenu de toutes ces difficultés, nous nous sommes posé cette question qui constituait notre problématique : Quelle solution adoptée pour améliorer le processus de collecte de données de consommation d'eau auprès des abonnés de la Regideso ?

Eu égard à cette problématique, nous avions émis l'hypothèse selon laquelle la solution pouvant prendre en compte la mobilité des agents au terrain pour intégration solide avec le reste de l'infrastructure logicielle serait l'implémentation d'une nouvelle couche logicielle mobile sous Android reposant sur la technologie REST facilitant les échanges entre les agents et le serveur d'application qui est hébergé au sein de l'entreprise. Cette solution permettrait à la Regideso / Kisangani d'améliorer les opérations de prélèvement des indicateurs des compteurs auprès de ses abonnés et de traitement automatiquement des données issues des abonnés en vue de déclencher les opérations de facturation dans un délai jugé satisfaisant.

Pour mener à bien cette étude, nous avions recouru aux méthodes UP et COCOMO respectivement pour modéliser le système d'information et estimer le cout global de l'application ainsi développer. La technique documentaire et la technique d'interview nous ont servi d'appui dans la phase de collecte des données. Pour atteindre notre objectif que nous nous sommes fixés dans cette étude, nous avions subdivisé ce travail en quatre sections hormis l'introduction et la conclusion.

La première section qui s'est intéressé à la définition des concepts de base en vue d'éclairer le lecteur sur la terminologie et les architectures utilisées. La seconde section consistait à faire l'étude fonctionnelle et technique de l'application à mettre en place en fixant les fonctionnalités nécessaires attendues et à poser les bases technologiques de l'ensemble de l'architecture logicielle à construire. Dans cette section, nous avions également fait usage de la méthode COCOMO qui nous a permis d'estimer le cout de cette réalisation informatique. La troisième section abordait la conception du nouveau système en faisant usage de la méthode UP utilisant le langage UML pour capturer les besoins utilisateurs par la présentation de séquence et de classes. La dernière section quant à elle s'est basée sur la présentation des technologies indispensable à la mise en œuvre de cette application.

Les différentes étapes définies dans cette étude ont constitué le cadre idéal pour développer une application informatique distribuée composée des plusieurs couches logicielles. Cette démarche ainsi adoptée permet à tout informaticien de se lancer dans la construction des applications reposant sur des architectures logicielles distribuées et facilite le choix des technologies à adopter. Compte tenu de cette clarification et de cette démarche, nous pouvons en toute humilité confirmer notre hypothèse et affirmer haut que l'objectif que nous nous sommes fixés a été atteint.

Ainsi nous suggérons aux responsables de la Regideso de Kisangani d'intégrer cet outil dans le processus de collecte de données et de traitement lié à la facturation de consommation d'eau des abonnés pour tirer profit des avantages de l'informatisation. Ce travail étant une œuvre humaine, il est ne manque pas d'imperfections et des remarques, c'est ainsi que nous sollicitons la contribution de nos lecteurs pour leurs critiques constructives et sommes disposé à recevoir dans un esprit scientifique leurs suggestions pour améliorer d'avantage notre travail.

BIBLIOGRAPHIE

Ouvrages

[1] Antonio Goncalves, Java EE 6 et GlassFish 3, Pearson Education France, 2010, ISBN
[2] : 978-2-7440-4157-0.
[3] Fien VAN DER HEYDE et Laurent DEBRAUWER, UML 2 Initiation, exemples et exercices corrigés 2ième édition, Editions ENI.
[4] Frédéric Chuong, Olivier Corgeron Cyril Joui, Jean-Baptiste Renaux et Maxime Vialette, op. cit., p.2
[5] Gérard CASANOVA - Denis ABÉCASSIS, Gestion de projet - calculs des coûts, Université de Lorraine, 2014, p.11
[6] Goncalves A., les Cahiers du Programmeur : Java EE 5, éditions Eyrolles, 2007, ISBN
[7] : 978-2-212-12038-7
[8] Joseph Gabay et David Gabay, UML 2. Mise en œuvre guidée avec études de cas : ANALYSE ET CONCEPTION, Dunod, Paris, 2008
[9] Kenneth L., Jane L., Eric F., Serge C. et Sophie C., management des systèmes d'information, 13ième édition, Nouveaux Horizons, 2013.
[10] M.Bigaud, JP Bourey, H. Camus, D. Corbeel, conception des systèmes d'information : Modélisation de données, étude des cas, Edition TECHNIP, Paris, 2006, page 14.
[11] Pascal Roques et Franck Vallée, UML 2 en action, De l'analyse des besoins à la conception 4e édition, Eyrolles 2007.
[12] Pitman, N., *UML 2 en concentré.* O'Reilly 2006.
[13] Robert ENGLANDER, Java et SOAP, O'Reilly, 2009, page 45.
[14] Safae Laqrichi. Approche pour la construction de modèles d'estimation réaliste de l'effort/coût de projet dans un environnement incertain : application au domaine du développement logiciel, thèse de doctorat, Université de Toulouse, Informatique, délivré par l'Ecole des Mines d'Albi-Carmaux, 2015.
[15] Statut de la regideso.

WEBOGRAPHIE

[1] http://remy-manu.developpez.com/introjava2ee
[2] https://fr.wikipedia.org/wiki/Constructive_Cost_Model
[3] https://www.lecoindesjeux.com/rediger-une-specification-fonctionnelle-detaillee/