Ministère de l'Enseignement Supérieur, de la Recherche Scientifique et de l'Innovation (**M.E.S.R.S.I**)

Secrétariat Général (**S.G**)

Université Nazi BONI (**U.N.B.**)

Ecole Supérieure d'Informatique (**ESI**)

Cycle des Ingénieurs de Conception en Informatique (**CICI**)

N° 68

MEMOIRE DE FIN DE CYCLE

THEME : Développement d'un répertoire téléphonique intégrant les relations sociales

Stage effectué du 1er février au 30 juin 2017

Présenté par :

Ibraïma DAGNOGO et Rashid Ben Amed Charles ZONGO

Directeur de mémoire

Pr Théodore TAPSOBA

Enseignant-chercheur

Ecole Supérieure d'Informatique

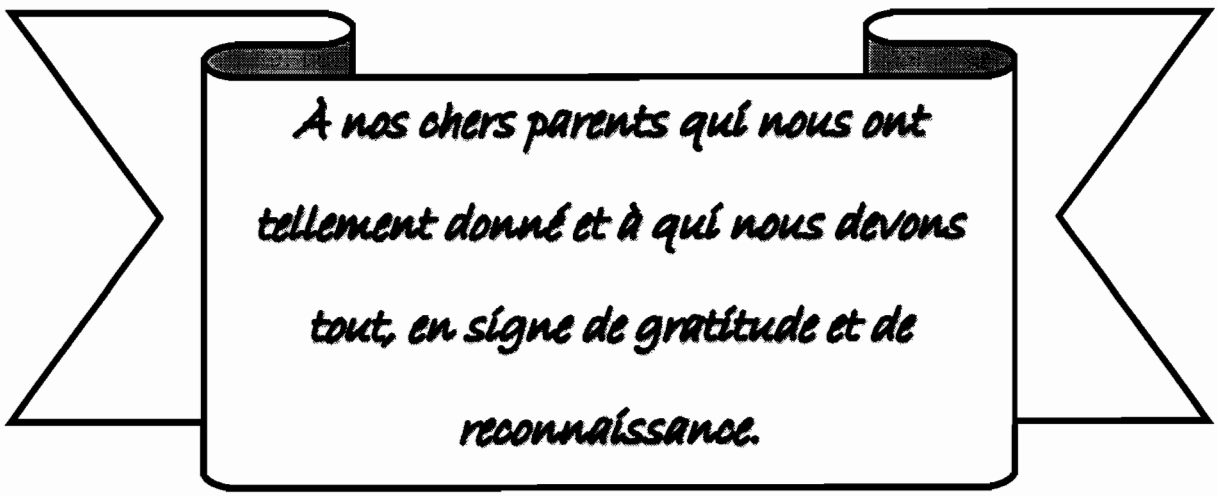
Co-directeur de mémoire

Dr Pasteur PODA

Enseignant-chercheur

Ecole Supérieure d'Informatique

DEDICACE


À nos chers parents qui nous ont
tellement donné et à qui nous devons
tout, en signe de gratitude et de
reconnaissance.

REMERCIEMENTS

Nos sincères remerciements s'adressent à :

- l'École Supérieure d'Informatique et à travers elle tout le corps enseignant et administratif pour la qualité de la formation ;
- Pr Théodore TAPSOBA, notre directeur de mémoire, pour nous avoir ouvert les portes de son laboratoire et aussi pour sa disponibilité, son apport et ses remarques pertinentes ;
- Dr Pasteur PODA, notre co-directeur de mémoire, initiateur de notre projet, pour sa disponibilité, son apport et ses remarques pertinentes ;
- Dr Borlli Michel SOME, membre de l'équipe dirigeante du projet, pour sa disponibilité et ses remarques pertinentes ;
- Dr Joelle OUATTARA, membre de l'équipe dirigeante du projet, pour sa disponibilité et ses remarques pertinentes ;
- nos camarades de classe et amis, pour l'amitié et la bonne collaboration ;
- nos frères et sœurs pour leur soutien.

SOMMAIRE

DEDICACE I
REMERCIEMENTS II
LISTE DES FIGURES VI
LISTE DES TABLEAUX VII
SIGLES ET ACRONYMES VIII
AVANT-PROPOS IX
INTRODUCTION GENERALE 1
CHAPITRE I : CADRE DE L'ETUDE 2
Introduction 3
I.1 Présentation de l'Ecole Supérieure d'Informatique 3
I.1.1 Filières de formation 3
I.1.2 Missions 3
I.1.3 Organisation 3
I.2 Présentation du Laboratoire d'Algèbre, de Mathématiques Discrètes et d'Informatique 4
I.3 Présentation du projet 4
I.3.1 Problématique 4
I.3.2 Résultats attendus 5
I.4 Gestion du projet 5
I.4.1 Acteurs du projet 5
I.4.1.1 Groupe de pilotage 5
I.4.1.2 Groupe de projet 6
I.4.1.3 Groupe des utilisateurs 6
I.4.2 Planning du projet 6
I.4.2.1 Détermination de la liste des tâches 6
I.4.2.2 Diagramme de GANTT 7
Conclusion 8
CHAPITRE II : ETUDE PRELIMINAIRE 9
Introduction 10
II.1 Etat des travaux et inventions relatifs aux listes de contacts 10
II.2 Recueil de fonctionnalités 11
II.3 Analyse des besoins techniques 11

II.3.1 Méthode de développement 11
II.3.2 Langage de modélisation 12
II.4 Outils de gestion des données 13
II.4.1 Technologies de l'intelligence artificielle classique 13
II.4.2 Technologies des données liées 14
II.4.2.1 Généralités sur le web de données 14
II.4.2.2 Technologies du web de données : ontologie 16
II.4.2.3 Technologies du web de données : langage de requêtes 17
II.4.2.4 Technologies du web de données : moteur d'inférences 18
Conclusion 19
CHAPITRE III : REPERTOIRE TELEPHONIQUE BASE SUR LES RELATIONS SOCIALES 20
Introduction 21
III.1 Modèle de répertoire téléphonique basé sur les relations sociales 21
III.2 Spécification des fonctionnalités 22
III.2.1 Acteurs du système 22
III.2.2 Cas d'utilisation 22
III.2.3 Description de quelques cas d'utilisation 23
III.2.4 Concepts du système 24
III.3 Modélisation du système 24
III.3.1 Diagramme de cas d'utilisation 24
III.3.2 Diagramme de séquences 25
III.3.3 Diagramme de classes 27
III.4 Etude algorithmique et calcul de complexité 28
III.4.1 Définition 28
III.4.2 Quelques algorithmes 28
III.4.3 Calcul de complexité : complexité temporelle 29
III.4.3.1 Définition 29
III.4.3.2 Notation $O(.)$ 30
III.4.3.3 Algorithme de recherche de liens 30
III.4.3.4 Algorithme de recherche par liens 31
III.4.3.5 Algorithme de résolution d'homonymie 31
Conclusion 32
CHAPITRE IV : IMPLEMENTATION DU REPERTOIRE TELEPHONIQUE BASE SUR LES RELATIONS SOCIALES 33
Introduction 34
IV.1 Outils de développement 34

IV.1.1 Langage de développement 34
IV.1.2 Environnement de développement 35
IV.1.3 Construction d'ontologies avec Protégé 2000 35
IV.1.3.1 Composantes d'une ontologie 35
IV.1.3.2 Classes de l'ontologie 36
IV.1.3.3 Propriétés 37
IV.1.4 Architecture du système 39
IV.1.4.1 Architecture technique des composantes 39
IV.1.4.2 Architecture physique des données 40
IV.1.5 Comparaison entre les différentes technologies utilisées 43
IV.1.5.1 Réalisation : OntoRep 44
Conclusion 48
CHAPITRE V : VERS UN REPERTOIRE TELEPHONIQUE INTELLIGENT 49
Introduction 50
V.1 Fonctionnalités intelligentes 50
V.1.1 Communication de groupes 50
V.1.1.1 Description de la fonctionnalité 50
V.1.1.2 Présentation de captures d'écran 50
V.1.2 Système de recommandations 53
V.1.2.1 Description de la fonctionnalité 53
V.1.2.2 Présentation de captures d'écran 53
Conclusion 54
CONCLUSION GENERALE 55
REFERENCES BIBLIOGRAPHIQUES 56
ANNEXES XI
- Règles de calcul de complexité XI
- JPL XII
- JENA XIII

LISTE DES FIGURES

Figure 1. Diagramme de Gantt du projet 7
Figure 2. Connexion entre Java et SWI-Prolog via JPL 14
Figure 3. Architecture du web de données 15
Figure 4. Graphe de connaissances 21
Figure 5. Diagramme de cas d'utilisation 25
Figure 6. Diagramme de séquence du cas d'utilisation « Enregistrer un contact » 26
Figure 7. Diagramme de séquence du cas d'utilisation « Rechercher un contact » 27
Figure 8. Diagramme de classes 28
Figure 9. Vue partielle de la hiérarchie des concepts 37
Figure 10. Interface de gestion des propriétés d'objets 38
Figure 11. Interface de gestion des propriétés de données 39
Figure 12. Architecture du système : approche basée sur Prolog 39
Figure 13. Architecture du système : approche à base ontologique 40
Figure 14. Implantation physique des contacts 41
Figure 15. Vue partielle ontographe représentant les concepts du système 42
Figure 16. Interface d'accueil (A) et liste de contacts (B) 44
Figure 17. Profil homme (A) et profil femme (B) 45
Figure 18. Illustration du lien existant entre deux contacts : ganama haoua est la mère de zongo farida (A) et ouedraogo djeneba est la grand-mère de zongo farida (B) 45
Figure 19. Formulaire d'enregistrement de contacts : formulaire vide (A) et exemple de formulaire rempli (B) 46
Figure 20. Modification d'un contact : contact à modifier (A), champs à modifier (B) et contact modifié (C) 47
Figure 21. Recherche de contacts par nom (A) et recherche de contacts par lien (B) 47
Figure 22. Interface de gestion de la communication de groupes 51
Figure 23. Critères de choix (A) et membres de groupe (B) 52
Figure 24. Constitution de groupes avec le critère « tous les liens » 52
Figure 25. Définition de la date de naissance 54
Figure 26. Message de suggestion ou de rappel 54
Figure 27. Déclaration dans le path des chemins pour JPL XII
Figure 28. Création de la librairie « JPL » XIII
Figure 29 : Création de la librairie « Jena » XIV
Figure 30. Ajout de la librairie « Jena » dans un projet JAVA XIV

LISTE DES TABLEAUX

Tableau 1. Liste des tâches 6
Tableau 2. Etude comparative de MERISE et UML 12
Tableau 3. Liens sociaux significatifs 22
Tableau 4. Liste des cas d'utilisation 22
Tableau 5. Description textuelle du cas d'utilisation « Enregistrer un contact » 23
Tableau 6. Description du cas d'utilisation « Rechercher un contact » 24
Tableau 7. Complexité : recherche de liens 30
Tableau 8. Complexité : recherche par liens 31
Tableau 9. Complexité : Résolution d'homonymie 32
Tableau 10. Etude comparative des langages de développement 34
Tableau 11. Etude comparative de quelques EDI 35
Tableau 12. Comparaison entre technologies classiques et technologies des données liées .. 43

SIGLES ET ACRONYMES

API : Application Programming Interface

CFR : Centre de Formation et de Recherche

CSAF : Chef de Service Administratif, Financier

DOE : Differential Ontology Editor

EDI : Environnement de Développement Intégré

ESI : Ecole Supérieure d'Informatique

HSQLDB : Hyper SQL Data Base

IA : Intelligence Artificielle

ITQL : Interactive Tucana Query Language

JESS : Java Expert System Shell

JNI : Java Native Interface

JPL : Java Prolog Library

LAMDI : Laboratoire d'Algèbre, de Mathématiques Discrètes et d'Informatique

OWL : Ontology Web Language

PROLOG : PROgrammation LOGique

RDF : Resource Description Framework

RDF-S : Resource Description Framework Schema

SPARQL : SPARQL Protocol And RDF Query Language

SQL : Structured Query Language

UML : Unified Modeling Language

UNB : Université Nazi BONI

UP : Unified Process

URI : Uniform Resource Identifier

W3C : World Wide Web Consortium

XML : eXtensible Markup Language

AVANT-PROPOS

Dans le but de décentraliser la formation universitaire qui était centrée à l'Université de Ouagadougou (UO), un centre universitaire fut créé en 1995 à Bobo-Dioulasso. Ce centre devient une université en 1997 sous le nom de «Université Polytechnique de Bobo-Dioulasso (UPB) ». Le 09 mai 2017, elle changea de nom et devient « Université Nazi BONI ». En effet, c'est un établissement public à caractère scientifique et technique, chargé d'enseignement supérieur et de recherche scientifique. Elle comprend :

- l'Ecole Supérieure d'Informatique (ESI) ;
- les instituts:
  - l'Institut de Développement Rural (IDR) ;
  - l'Institut Nationale des Sciences de la Santé (INSSA) ;
  - l'Institut Universitaire de Technologie (IUT) ;
- les Unités de Formations et de Recherches:
  - Science Juridique et Politique (SJP) ;
  - Science Juridique Politique Economique et Gestion (SJPEG) ;
  - Science et Technologie (Génie Biologie, Maths Informatique, Science Biologie) ;

L'Ecole Supérieure d'Informatique (ESI) a pour mission d'accompagner le pays dans son ambition de s'approprier les technologies de l'information et de la communication. Ainsi elle forme en cinq ans des ingénieurs de conception en informatique. Pour évaluer les connaissances plus ou moins théoriques des enseignements et permettre une bonne intégration des étudiants sur le marché de l'emploi, un stage pratique de fin de cycle est institué.

D'une durée d'au moins quatre mois, ce stage donne l'occasion aux étudiants de réaliser un projet de fin d'étude à l'issue duquel une soutenance publique est organisée. Ce projet consiste en la résolution d'un problème scientifique et technique par l'étudiant en faisant intervenir tout ou une partie de l'ensemble des disciplines relevant des sciences de l'ingénierie enseignées au cours de sa formation.

C'est ainsi que nous avons travaillé sur le thème : « **Développement d'un répertoire téléphonique intégrant les relations sociales ».**

INTRODUCTION GENERALE

Le téléphone portable, également appelé téléphone mobile ou téléphone cellulaire, a révolutionné notre vie quotidienne en nous permettant de communiquer sans fil, partout où il y a un réseau de téléphonie mobile. Ce téléphone mobile nous offre une panoplie de fonctionnalités. Une très intéressante fonctionnalité de base est le répertoire téléphonique. Le répertoire téléphonique sert à sauvegarder les noms et les adresses des contacts du propriétaire du téléphone mobile. Il permet au propriétaire d'appeler ses contacts sans être contraint de retenir leurs numéros. De nos jours, les téléphones portables offrent des répertoires avec des fonctions bien élaborées comprenant une diversité de champs qui servent à identifier et à se rappeler d'un contact enregistré. Certains de ces champs (société, note, surnom, etc.) ont pour but de donner plus d'informations sur un contact. Nous avons des répertoires contenant plusieurs centaines de contacts et même avec ces multiples champs, il devient compliqué de différencier des contacts de même nom, voire de se rappeler de certains contacts. De plus, nous avons besoin souvent de partager une certaine information avec une catégorie de personnes de notre répertoire. Il convient donc de trouver une solution assez simple qui, lors de l'enregistrement, permet de différencier les contacts et de se rappeler à coup sûr de chaque contact. Aussi, la solution devra notamment permettre de constituer des groupes de contacts en fonction des relations et de nos besoins. Il serait aussi très intéressant que le répertoire téléphonique puisse faire un certain nombre de recommandations à son utilisateur. C'est dans cette optique, qu'il a été pensé de prendre, d'une part, en compte les liens sociaux qui existent entre les contacts lors de leur enregistrement et d'autre part faire des recommandations à propos des contacts enregistrés selon des critères définis.

Ainsi ce rapport résume notre travail en cinq chapitres. Dans le premier chapitre, nous présentons le cadre de l'étude. Dans le deuxième chapitre, nous abordons les différents travaux déjà réalisés dans le domaine des listes de contacts en lien avec les relations et les recommandations. Au troisième, nous présentons une conception du modèle de répertoire téléphonique inspiré des pratiques et des comportements sociaux africains. Dans le quatrième chapitre, nous abordons l'implémentation de la nouvelle vision du répertoire téléphonique. Au cinquième chapitre, nous présentons l'aspect intelligent du modèle de répertoire téléphonique proposé.

1

CHAPITRE I : CADRE DE L'ETUDE

Introduction

Dans ce chapitre, il est question de présenter le cadre de l'étude et l'environnement dans lequel elle s'est déroulée. Il présente aussi le projet en faisant ressortir les acteurs et le planning prévisionnel.

I.1 Présentation de l'Ecole Supérieure d'Informatique

I.1.1 Filières de formation

L'École Supérieure d'Informatique (ESI), créée en 1991 est une école de l'Université Nazi BONI. On y distingue deux cycles de formation :

- Le Cycle des Ingénieurs de Travaux Informatiques (CITI) avec les options : Analyse et Programmation (AP) et Réseaux et Maintenance Informatiques (RéMI)
- Le Cycle des Ingénieurs de Conception en Informatique (CICI).

Depuis l'année 2014-2015, l'ESI a mis fin aux deux options AP et RéMI et adopté une politique de tronc commun de la première à la deuxième année. A partir de la troisième année, deux filières sont à la disposition des étudiants : « les systèmes d'information » et « les réseaux et systèmes ». Elle a aussi en projet, l'ouverture d'un master pour les deux filières citées préalablement qui remplacera le CICI.

I.1.2 Missions

La mission première de l'ESI est d'accompagner le pays dans son ambition de s'approprier les Technologies de l'Information et de la Communication (TIC). Pour ce faire, elle assure la formation fondamentale, appliquée et/ou professionnelle dans les domaines de l'informatique, la formation continue, la préparation des diplômés et la participation à des programmes internationaux de formation et de recherche.

I.1.3 Organisation

L'ESI est structurée en une direction et services selon l'arrêté ministériel n°2003-063/MESSRS/SG/UPB/R du 1er avril 2003 :

- le directeur : La direction de l'ESI est assurée par un directeur nommé par arrêté ministériel sur proposition du président de l'université Nazi BONI, ex Université polytechnique de Bobo-Dioulasso, après élection par un collège électoral.

- le directeur adjoint : le directeur adjoint est nommé dans les mêmes conditions que le directeur. Il assiste le directeur dans l'exercice de ses fonctions.
- le service d'enseignement : le corps enseignant de l'ESI est constitué d'enseignants chercheurs dans plusieurs domaines de l'informatique et aussi de professionnels très qualifiés.
- le service des stages : ce service gère tout ce qui est lié à la question des stages et l'organisation des soutenances.
- le service administratif, financier et comptable : ce service est chargé de la coordination de la gestion administrative et financière.
- la bibliothèque : elle est chargée de l'acquisition et la conservation d'ouvrages scientifiques et pédagogiques, des résultats et travaux de recherches, des thèses des enseignants-chercheurs et des mémoires, des rapports de stage des étudiants.
- le secrétariat.

Les enseignants chercheurs effectuent de nombreuses études dans le cadre de leurs recherches. La plupart du temps, ils font appel à des étudiants en fin de cycle pour travailler sur les thématiques de leurs recherches. C'est dans cette idée que s'inscrit notre stage de fin de cycle des ingénieurs de conception en informatique. Pour les travaux de ce stage, nous avons été accueillis au Laboratoire d'Algèbre, de Mathématiques Discrètes et d'Informatique (LAMDI).

I.2 Présentation du Laboratoire d'Algèbre, de Mathématiques Discrètes et d'Informatique

Situé dans l'enceinte du Centre de Formation et de Recherche (CFR) de l'Université Nazi BONI, le LAMDI, dirigé par Pr Théodore TAPSOBA, accueille des activités de recherches dans les domaines des mathématiques et de l'informatique. Il est composé de plusieurs équipes de recherche qui travaillent sur des thématiques de thèses, de mémoires de fin de cycle et de masters.

I.3 Présentation du projet

I.3.1 Problématique

Les listes de contacts ou les répertoires téléphoniques des équipements mobiles (smartphones, téléphones portables, etc.) peuvent facilement stocker aujourd'hui plusieurs centaines, voire des milliers de contacts. Telles qu'elles sont conçues jusqu'à présent, les listes de contacts ne prennent pas efficacement en compte les connexions sociales qui existent entre les individus. Pourtant, ce sont les connexions sociales qui sont les principales pourvoyeuses d'entrées dans

les listes de contacts. Aussi, les listes de contacts actuelles ne permettent pas de retrouver ou de se souvenir facilement d'un contact qu'on a oublié ou qui est concerné par l'homonymie. Une nouvelle vision de la liste de contacts sera de la concevoir de sorte qu'elle permette de mémoriser les liens sociaux qui existent entre ses différentes entrées. Au-delà, les liens sociaux pourront donner naissance à de nouveaux signes sur la base desquels des recommandations prendront place au sein du téléphone mobile. Autrement dit, la nouvelle vision de la liste de contacts permettra aussi de doter le téléphone de fonctionnalités intelligentes.

I.3.2 Résultats attendus

L'objet de cette étude est de parvenir à une implémentation d'un modèle de liste de contacts qui intègre les connexions sociales. Au-delà, l'objectif ultime est de se baser sur les liens sociaux pour développer des fonctionnalités intelligentes. Ainsi la nouvelle vision du répertoire téléphonique devra être capable :

- d'établir automatiquement des liens sémantiques entre les contacts ;
- de gérer efficacement l'homonymie entre les contacts ;
- de gérer efficacement les groupes de contacts ;
- de suggérer des actions envers des contacts ou suggérer des contacts.

I.4 Gestion du projet

I.4.1 Acteurs du projet

Les acteurs d'un projet sont toutes les personnes qui interviennent dans sa gestion. Ils sont divisés en trois groupes qui sont : le groupe de pilotage, le groupe de projet et le groupe des utilisateurs.

I.4.1.1 Groupe de pilotage

Le groupe de pilotage est un groupe d'encadreurs chargé d'arbitrer et de contrôler les décisions à prendre. Il valide les grands choix techniques, fixe les orientations générales et les délais à respecter. Il définit également les moyens à mettre en place pour la réalisation du projet et approuve le plan d'action établi par le groupe de projet.
Il est constitué de :

- **Dr Pasteur PODA**, enseignant-chercheur à l'ESI ;
- **Dr Borlli J. SOME**, enseignant-chercheur à l'ESI ;
- **Dr Joëlle OUATTARA**, enseignant-chercheur à l'ESI ;

- **Pr Théodore TAPSOBA,** enseignant-chercheur à l'ESI ;

I.4.1.2 Groupe de projet

Le groupe de projet est chargé de l'exécution du projet, c'est-à-dire l'analyse, la conception, la réalisation et le déploiement de l'application. Il établit également les rapports sur l'activité et l'avancement du projet auprès du groupe de pilotage.

Ce groupe est composé de :

- **Ibraïma DAGNOGO,** élève-ingénieur en Cycle des Ingénieurs de Conception en Informatique à l'ESI ;
- **Rashid Ben Amed Charles ZONGO,** élève-ingénieur en Cycle des Ingénieurs de Conception en Informatique à l'ESI.

I.4.1.3 Groupe des utilisateurs

Il est constitué des utilisateurs potentiels du système qui sera développé c'est-à-dire tout utilisateur de smartphones.

I.4.2 Planning du projet

La réalisation d'un projet passe par l'établissement et surtout le respect d'un planning bien défini en accord avec le groupe de pilotage. Ce planning doit tenir compte des contraintes liées à l'organisation du projet, du temps qui est imparti au groupe de projet et de la méthode d'analyse. Il doit permettre au groupe de projet de suivre l'avancement du projet.

I.4.2.1 Détermination de la liste des tâches

Afin de mieux gérer la complexité d'un travail, il convient de le diviser en de petites tâches plus faciles à comprendre et à réaliser. La liste des tâches est consignée dans le tableau 1.

Tableau 1. Liste des tâches

Tâches	Description
Revue bibliographique	Lecture des articles scientifiques concernant les différents travaux réalisés dans le domaine des listes de contacts.
Compréhension du sujet	Compréhension approfondie du sujet en délimitant le périmètre du projet

Analyse fonctionnelle	Détermination des différentes fonctionnalités de l'application.
Prototype	Réalisation d'un prototype.
Etudes des technologies	Détermination des technologies candidates dans la réalisation de notre projet.
Prise en main des technologies des données liées	Utilisation des technologies des données liées ou linked data.
Réalisation de l'application	Développement de l'application en utilisant les technologies choisies.
Rédaction du mémoire	Etape de rédaction du mémoire

I.4.2.2 Diagramme de GANTT

Le diagramme de GANTT est un outil utilisé en ordonnancement et en gestion de projet. Il permet de visualiser dans le temps les diverses tâches liées composant un projet. Il s'agit donc d'une représentation sous forme de graphe.

On remarquera que seuls les jours ouvrables ont été considérés dans l'élaboration du diagramme de GANTT illustré par la figure 1.

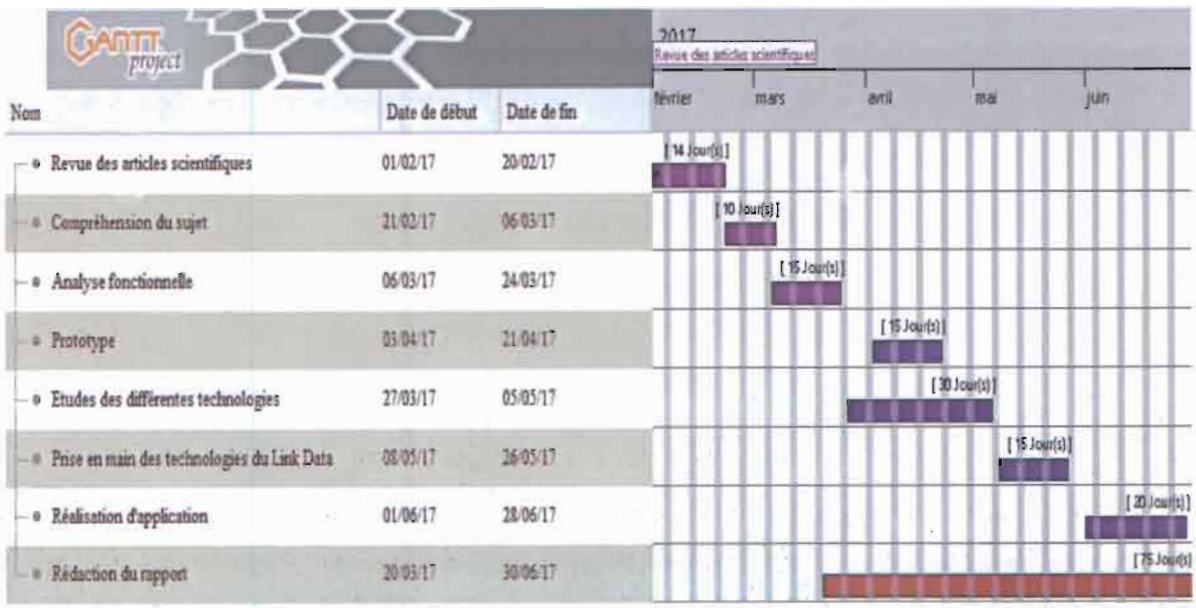


Figure 1. Diagramme de Gantt du projet

Conclusion

Ce chapitre aborde le cadre dans lequel nous avons effectué notre stage de fin de cycle. En effet il présente d'une part l'Ecole Supérieure d'Informatique et le LAMDI situé au Centre de Formations et de Recherches de l'Université Nazi BONI de Bobo-Dioulasso et d'autre part les acteurs du projet et le planning du projet. Afin d'atteindre les résultats attendus, une étude préliminaire a été effectuée.

2

CHAPITRE II : ETUDE PRELIMINAIRE

Introduction

Dans le cadre des études préliminaires de ce projet, un certain nombre d'articles scientifiques, abordant les aspects de répertoire téléphonique, de recommandations et de gestions des liens sociaux entre individus, ont été lus. Ce chapitre présente d'une part, un résumé de ces articles ainsi qu'un bilan issu de lecture, d'autre part une approche pour la résolution du problème et les outils de gestion de données.

II.1 Etat des travaux et inventions relatifs aux listes de contacts

Dans [1], l'idée est de créer un répertoire téléphonique intelligent qui permettra de faire des recommandations de contacts en se basant sur la situation courante de l'utilisateur. En effet, les situations de l'utilisateur dérivent des historiques en utilisant le réseau bayésien. Les historiques proviennent des opérations effectuées par l'utilisateur sur les appareils où ils sont stockés. La méthode proposée, en se servant des historiques collectés sur les mobiles, effectue un prétraitement avant que ces historiques ne soient utilisés pour le réseau bayésien. Afin de recommander un contact en lien avec la situation actuelle de l'utilisateur, une comparaison est faite entre la situation courante et l'historique stockée sur l'appareil. Lorsque des similarités sont trouvées entre le contexte actuel et une situation de l'historique, la méthode calcule un score afin de déterminer la situation la plus évidente (score le plus élevé) et la propose à l'utilisateur.

Dans [2], l'auteur a proposé un algorithme basé sur le système de contextes. En effet l'algorithme compare la notion de similarité incluant les informations sur le qui, le quand, le où, le quoi et les images, puis donne des informations en se basant sur l'analyse globale de la comparaison des informations personnelles et des informations de contexte du réseau social.

Afin de découvrir et de cibler les besoins des utilisateurs aux moments opportuns dans un réseau social, l'article [3], propose une solution qui utilise les relations sociales et un certain nombre d'informations pour recommander des résultats en lien avec les besoins de l'utilisateur. Cet article présente un algorithme de recommandations pour les requêtes des utilisateurs basées sur les réseaux sociaux, qui combine les profils des utilisateurs, leurs relations sociales, et des informations liées au contexte pour faire une recommandation ciblée pour la demande de l'utilisateur. Grâce à cet algorithme de recommandations, les utilisateurs peuvent rapidement localiser et avec précision les informations dont ils ont besoin.

En s'intéressant aux réseaux sociaux et à la modélisation des contextes sociaux, Lee et al [4] ont développé un service de transformation d'émotions en émoticons. Ce service déduit les

émotions de l'utilisateur et la transfert aux autres utilisateurs du réseau social sous forme d'émoticons.

On peut remarquer que [1], [2] et [4] mettent plus d'attention sur la modélisation du contexte et les émotions, sans tenir compte des relations sociales. Il est à noter que, bien que [1] a mentionné l'utilisation d'informations sociales pour concevoir des algorithmes et recommander des notes, il n'implique pas la relation d'amis. Dans l'article [3], l'accent est mis sur les relations d'amitié dans les réseaux sociaux. Il se limite juste à cette relation amicale et ne fait pas ressortir d'autres connexions sociales plus spécifiques qui peuvent exister entre les individus.

II.2 Recueil de fonctionnalités

Le nouveau répertoire téléphonique propose en plus des fonctionnalités d'un répertoire téléphonique classique, les fonctionnalités suivantes :

- la gestion des liens sociaux : création de liens, déduction de relations, recherche par liens ;
- la gestion de groupes de contacts : constitution de groupes ;
- la gestion de l'homonymie : différenciation des contacts par leurs relations ;
- la gestion de recommandations : suggestion de contacts.

II.3 Analyse des besoins techniques

II.3.1 Méthode de développement

En ingénierie logicielle, une méthode d'analyse et de conception ou méthode de développement est un procédé qui a pour objectif de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client. Il existe plusieurs types de méthode de développement, chacune avec ces spécificités tels que : Unified Process Two Tracks Unified Process, Cascade, Xtreme Programming.

Dans le cadre de notre projet, nous avons opté pour la méthode Unified Process (UP) ou processus unifié. Le processus unifié regroupe les activités à mener pour transformer les besoins des utilisateurs en un système logiciel. Ses caractéristiques essentielles sont qu'il :

- est à base de composants ;
- utilise le langage UML (ensemble d'outils et de diagrammes) ;
- est piloté par les cas d'utilisation ;
- est centré sur l'architecture ;

- est itératif et incrémental.

Ces différentes caractéristiques correspondent assez bien à notre projet.

II.3.2 Langage de modélisation

Un langage de modélisation est un langage artificiel qui peut être utilisé pour exprimer de l'information ou de la connaissance ou des systèmes dans une structure qui est définie par un ensemble cohérent de règles. Ces règles sont utilisées pour l'interprétation de la signification des composants dans la structure.

Pour la modélisation des systèmes d'informations, les concepteurs ont recours généralement a deux (02) approches: UML et MERISE. A travers le tableau suivant, nous menons une étude comparative de ces deux (02) approches afin de choisir le plus adapter à notre contexte comme l'indique le tableau 2.

Tableau 2. Etude comparative de MERISE et UML

Langage de modélisation	Points forts	Points faibles
UML	– Langage formel et normalisé ; – Support de communication performant ; – Compréhension facile des représentations abstraites complexes ; – Souple, polyvalent et universel ; – Etc.	– Absence de méthodologie ;
MERISE	– Approche systémique ; – Concepts simples et peu nombreux ; – Indépendant vis-à-vis de la technologie ;	– Limité au niveau organisationnel ; – Ne permet pas une validation rapide de la part des utilisateurs ; – Difficile de valider les traitements par rapport aux données ;

Suite à l'étude précédente, nous choisissons UML comme langage de modélisation de notre système, car il comble une lacune importante des technologies objets. Il permet d'exprimer et d'élaborer des modèles objets, indépendamment de tout langage de programmation. De plus, grâce à sa notation graphique, il permet d'exprimer visuellement une solution objet, ce qui

facilite la comparaison et l'évaluation de solutions. Enfin, l'aspect formel de sa notation limite les ambiguïtés et les incompréhensions.

II.4 Outils de gestion des données

Dans notre projet, nous devons mettre en place des techniques qui nous permettront de bien gérer les données liées aux contacts. Cette gestion des données doit être en mesure de déduire de nouvelles données à partir des données de base. A cet effet nos données doivent être des faits ou des connaissances et doivent se baser sur des règles pour inférer d'autres faits. Pour réaliser cette gestion des données, il nous faut :

- une base de connaissances ou une base de faits ;
- un ensemble de règles ;
- un moteur d'inférences, sous-système qui permet d'appliquer les règles à la base de faits ou la base de connaissances.

II.4.1 Technologies de l'intelligence artificielle classique

L'outil le mieux adapté pour mettre en place une telle gestion est Prolog, acronyme de PROgrammation LOGique. C'est un langage de programmation logique parfaitement adapté à la réalisation d'une base de données déductive. Il a été conçu pour des systèmes simulant l'intelligence humaine. En effet avec Prolog, il ne sera stocké que l'essentiel des données c'est-à-dire que les faits de base, le reste étant de simples déductions ; ce qui est très bénéfique en termes de mémoire de stockage. Afin de pouvoir mettre en place un tel système de stockage, nous utilisons SWI-Prolog par l'intermédiaire duquel nous concevons les fichiers de notre base de données.

Pour la gestion des données, nous utilisons des fichiers Prolog. Notre application se développant sous le langage JAVA, il faudrait donc trouver un moyen de faire communiquer notre système avec les fichiers de la base de faits. Pour ce faire, il faut une interface entre le langage Java et Prolog par l'intermédiaire de SWI-Prolog. L'API JPL est la mieux adaptée pour servir d'interface. JPL est un ensemble de classes Java et de fonctions C fournissant une interface entre Java et Prolog. JPL utilise Java Native Interface (JNI) pour se connecter à un moteur Prolog via l'interface linguistique Prolog. Elle n'est pas une application Java pure de Prolog. La version actuelle de JPL ne fonctionne qu'avec SWI-Prolog. En effet elle permet aux applications Prolog d'exploiter les classes Java, les instances, les méthodes, etc. Aussi elle autorise les applications Java à manipuler toutes les bibliothèques Prolog standard, les prédicats,

etc. Enfin elle autorise les applications hybrides Prolog + Java à être conçues et mises en œuvre de manière à tirer le meilleur parti des deux systèmes linguistiques [8]. La figure 2 présente la connexion entre Java et SWI-Prolog via JPL.

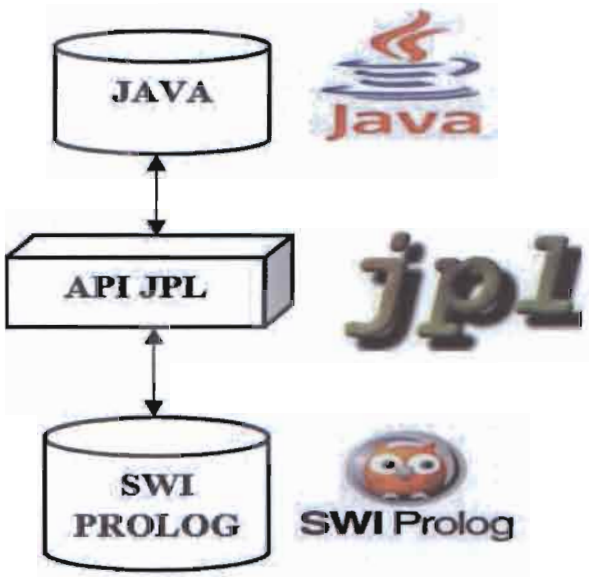


Figure 2. Connexion entre Java et SWI-Prolog via JPL

II.4.2 Technologies des données liées

II.4.2.1 Généralités sur le web de données

Le web de données ou données liées (en anglais linked data) désignent la mise sur le web des données mais surtout leur mise en relation pour constituer un réseau global de données, où, à partir d'une donnée, on accède aux autres données liées sur le web. Afin de permettre cette mise en relation des données, le web de données structure les données sous forme de triplets. Ces triplets sont composés des éléments suivants :

- un **sujet** : en général une instance, c'est l'élément décrit par le triplet ;
- un **prédicat** : représente un type de propriété applicable au sujet ;
- un **objet** : peut-être une autre instance ou un littéral.

Pour parvenir à une telle structuration des données sur le web, il faut utiliser des standards et des technologies définis par le World Wide Web Consortium (W3C), qui sont résumés dans la figure 3 [9].

En effet l'ensemble des technologies du Web sémantique est organisé dans une architecture en couches. C'est ce qu'on appelle en anglais la « Semantic Web Stack » et qui se traduit en français « Pile du Web Sémantique ».

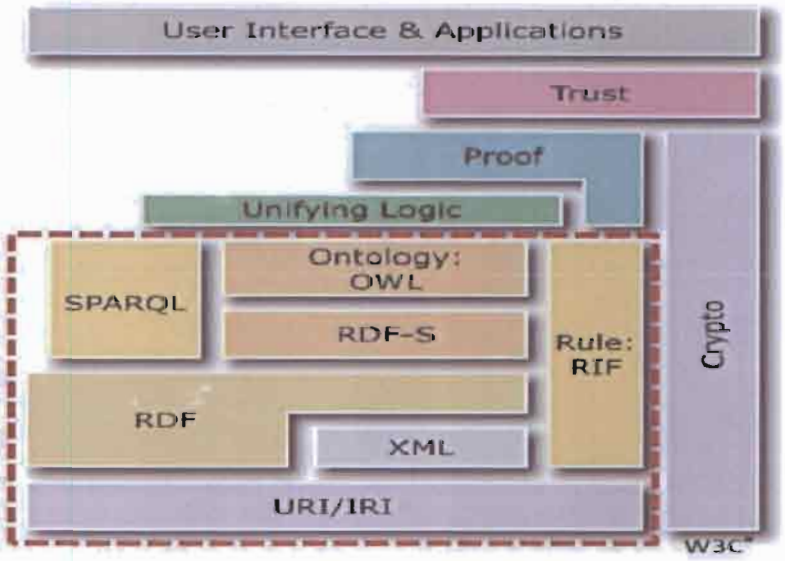


Figure 3. Architecture du web de données

Chaque couche a une fonction bien définie et précise [15] :

- la couche URI (Uniform Resource Identifier) : c'est un protocole simple et extensible pour identifier, d'une manière unique et uniforme, toute ressource sur le web ;
- la couche XML : il s'agit d'une couche syntaxique, de bas niveau, qui permet de structurer et organiser les données selon un format de message standard, et ce, grâce au langage de balisage extensible XML ;
- les couche RDF et RDFS : après avoir référencé les ressources avec le protocole URI et structurées les informations avec le XML, I 'étape suivante consiste à les annoter (affecter des notes : métadonnées), afin de les doter d'un sens interprétable par la machine. C'est justement le rôle des couches RDF et RDFS dans l'architecture du Web sémantique.
- La couche ontologique : il s'agit implicitement des langages ontologiques, dont le plus connu est l'OWL. C'est un vocabulaire XML basé sur le RDF qui permet d'élaborer des ontologies web structurées ;
- La couche de requêtes : elle aborde le langage SPARQL. C'est un langage de requêtes et un protocole qui permettra de rechercher, d'ajouter, de modifier ou de supprimer des données RDF disponibles dans le web à travers l'Internet.

- La couche de RIF (Rule Interchange Format) : c'est la couche des règles dans l'architecture du web sémantique ;
- La couche logique : cette couche utilise les règles pour déduire de nouvelles informations ;
- La couche preuve : elle a pour but de prouver la pertinence de l'information retournée par les couches de plus bas niveau et des déductions obtenues à partir des inférences ;
- La couche confiance : elle a pour objectif d'évaluer la fiabilité de l'information et des raisonnements ;
- La couche de cryptographie : elle est utilisée ici afin de s'assurer de la véracité et de la fiabilité de sources fournissant des données dans le cadre du Web sémantique ;
- La couche interface utilisateur : c'est le dernier élément de la pyramide du web sémantique permettant d'utiliser les applications utilisant le web sémantique.

II.4.2.2 Technologies du web de données : ontologie

Une ontologie permet de structurer un ensemble de concepts afin de leur donner un sens. Elle définit notamment l'ensemble de rapports possibles entre ces concepts et permet de créer des restrictions et des conditions sur ces liens. Une ontologie peut être utilisée pour déduire de nouvelles informations à partir des informations disponibles [11]. Elle est utilisée dans le web sémantique, dans la gestion des connaissances, dans l'extraction de données, etc.

Afin d'établir une norme permettant aux différentes applications utilisant le web de données liées de se comprendre, le W3C a publié plusieurs langages permettant de standardiser la structure des ressources [7]. Parmi ceux-ci, on distingue deux types de langages :

- les langages comme RDF (Resource Description Framework) et RDFS (RDF Schema) permettant de décrire des données ;
- Les langages comme OWL permettant de structurer les données.

Le langage RDF est à la base du web sémantique. La syntaxe principale utilisée par RDF est RDF/XML. RDF sert de base aux langages plus complexes du web sémantique notamment, les langages d'ontologie tels que RDFS et OWL.

RDFS est une extension du langage RDF. Il permet de décrire des ontologies simples en ajoutant des notions de classe, de sous-classe, de sous propriété, de domaine et de portée. RDFS est inclus dans le langage d'ontologie OWL.

OWL permet de définir des ontologies structurées. En plus des concepts de RDF et RDFS, OWL ajoute les concepts de classes équivalentes, de propriétés équivalentes, d'égalité de deux

ressources, de leurs différences, du contraire, de symétrie et de cardinalité. OWL définit trois sous-langages différents :

- OWL/Lite : est la version la moins expressive de OWL ;
- OWL/DL : version plus complète, elle est utilisée par les raisonneurs tels que Racer ou Pellet ;
- OWL/Full : version la plus complète, elle est destinée aux utilisateurs désirant une description très précise des concepts bien qu'aucun raisonneur ne la supporte à l'heure actuelle [12].

L'une des principales décisions à prendre dans le procédé de développement des ontologies consiste à choisir le langage dans lequel l'ontologie sera exprimée et utilisée. Pour nos besoins, nous avons opté pour le langage OWL dans sa version OWL/DL pour les raisons suivantes :

- OWL/Lite ne permet d'exprimer que des contraintes simples de cardinalité 0 ou 1, tandis que OWL/DL permet d'exprimer des cardinalités multiples.
- OWL/Full offre un plus haut niveau d'expressivité que demande l'ontologie de ce projet, tandis qu'OWL/DL offre un niveau d'expressivité suffisant.

Pour la création des ontologies, nous pouvons utiliser un éditeur d'ontologies. Celui-ci génèrera le code de l'ontologie dans le langage choisi, dans notre cas en OWL. De nos jours, il existe de nombreux outils de construction (édition et visualisation) qui utilisent des formalismes variés et offrent différentes fonctionnalités. Parmi ces outils, nous citons les quatre suivants : Protégé 2000, KAON, Ontolingua, DOE (Differential Ontology Editor) [6].

Pour la création de notre ontologie, nous avons opté pour l'éditeur Protégé 2000 dans sa version 5.0. En effet, cet outil, open source, disponible à l'adresse http://protege.standford.edu, développé au département d'Informatique Médicale de l'Université de Standford en Californie aux Etats-Unis, offre une interface graphique assez facile à maitriser et possède une importante communauté sur Internet.

II.4.2.3 Technologies du web de données : langage de requêtes

Un langage de requêtes est un langage informatique utilisé pour accéder aux données d'une base de données ou d'autres systèmes d'information. Il permet d'obtenir les données vérifiant certaines conditions. Un langage de requêtes est spécifique à un type de systèmes d'information. Il existe plusieurs types de langages de requêtes tels que le Datalog pour les bases de données

déductives, le SQL pour les bases de données relationnelles, le SPARQL pour les graphes RDF et XQuery pour les données XML [14].

Le langage de requêtes adéquat pour les technologies du web sémantique est SPARQL. Le langage SPARQL définit la syntaxe des requêtes effectuées sur un graphe de données RDF. Inspiré du langage SQL, il se base sur les triplets contenus dans le graphe. Il est composé de trois parties distinctes [14] :

- les PREFIX qui sont en fait les préfixes des URI des ressources à utiliser ;
- la clause SELECT, semblable à celle du langage SQL, permet de définir les différentes ressources retournées par la requête ;
- la clause WHERE, composée d'un ensemble de triplets, permet de définir les conditions dans la sélection.

II.4.2.4 Technologies du web de données : moteur d'inférences

Il existe plusieurs outils pour la gestion des données sémantiques. Comme outils, nous pouvons citer :

- Jena : un Framework sémantique Java possédant un riche panel de fonctionnalités et une importante communauté sur le web ;
- Mulgara : un système de stockage de données sémantiques ;
- HSQLDB : un système de bases de données embarquées.

Tous les outils appartiennent à des catégories différentes. Cependant, chacun d'entre eux comporte des fonctionnalités utiles. Dans le cadre de ce projet, nous avons opté pour Jena car il regroupe toutes les caractéristiques dont nous avons besoin.

Jena est un Framework Java destiné au développement d'applications du Web sémantique. Il offre des outils de programmation pour la gestion des standards tels que OWL, SPARQL et inclut un moteur d'inférences basé sur des règles de déduction. Jena est un projet open source [13]. Les principales fonctionnalités offertes par Jena sont :

- une API pour le standard RDF ;
- une API pour le standard OWL ;
- la lecture et l'écriture de données RDF dans les formats RDF/XML, N3 et N-Triples ;
- le stockage des ontologies en mémoire ou persistance ;
- un moteur de requêtes SPARQL ;

- un moteur d'inférences.

Conclusion

Ce chapitre a permis d'avoir une idée sur les différents travaux et recherches effectués dans le domaine des listes de contacts et des relations sociales. Il présente également une approche de résolution en décrivant la méthode de développement et aussi le langage de modélisation. Aussi, il aborde les différentes catégories et types de technologies intervenant dans la gestion des données.

3

CHAPITRE III : REPERTOIRE TELEPHONIQUE BASE SUR LES RELATIONS SOCIALES

Introduction

Le but de ce chapitre est de donner une vue d'ensemble du type de répertoire téléphonique à développer. La phase de conception décrira de manière précise et détaillée, le fonctionnement du système futur, afin d'en faciliter la réalisation. Il sera donc question de présenter le diagramme de cas d'utilisation, la description de quelques cas et les diagrammes de séquences.

III.1 Modèle de répertoire téléphonique basé sur les relations sociales

Traditionnellement, une liste de contacts mobiles est une base de données de contacts stockés dans un équipement mobile. Ces contacts stockés sont indépendants. La nouvelle liste de contacts proposée se démarque de cette indépendance des éléments de la liste de contacts. Dans ce nouveau système, les contacts sont liés entre eux par des relations sociales. En effet, ce répertoire peut être considéré comme un ensemble d'entités ou les entités sont reliées par des liens sociaux. La figure 4 illustre la structure du nouveau système proposé.

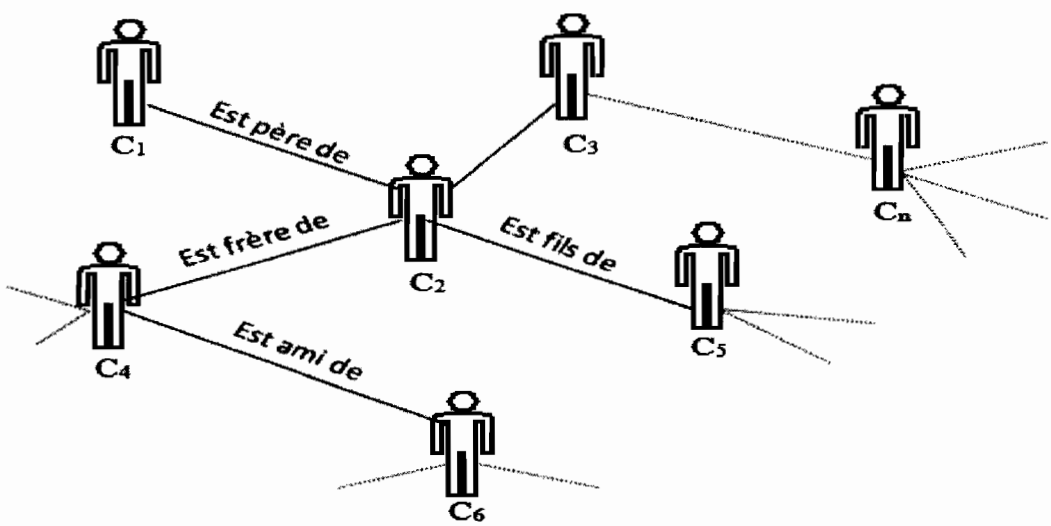


Figure 4. Graphe de connaissances

La spécificité majeure de ce modèle de répertoire se résume aux relations qui existent entre les contacts. Tout type de lien peut être envisagé. Cependant, les relations sociales significatives ont été retenues et regroupées en trois catégories comme le montre le tableau 3 [5].

Tableau 3. Liens sociaux significatifs

Catégories	Nominations
Liens familiaux	Est le père de, Est la mère de, Est le fils de, Est la fille de, Est le frère de, Est la sœur de, Est le cousin de, Est l'oncle de, Est la belle-sœur de, Est la femme de, Est le mari de, Est la tante de, ...
Liens amicaux	Est le camarade de classe de, Est l'ami de, Est le copain de, Est le camarade de,. . .
Liens professionnels	Est le collègue de, est le collaborateur de, Est le directeur de, Est le coiffeur de, ...

III.2 Spécification des fonctionnalités

III.2.1 Acteurs du système

Un acteur peut être un humain ou une machine ne faisant pas partie de la solution à réaliser mais qui participe au fonctionnement général de la solution par une interaction.
Définir les acteurs consiste donc à définir les limites du projet mais surtout les profils des différentes personnes ou machines qui vont interagir avec le système. Les acteurs susceptibles d'interagir avec notre système sont tout utilisateur de téléphone mobile.

III.2.2 Cas d'utilisation

L'identification des différentes fonctionnalités attendues permet de déterminer les cas d'utilisation dans le tableau 4.

Tableau 4. Liste des cas d'utilisation

Cas d'utilisation	Définition
Définir une relation	Permet de sélectionner une relation afin de lier deux contacts.
Enregistrer un contact	Permet de sauvegarder un contact dans la liste des contacts.
Lister les contacts	Permet d'afficher les différents contacts de la liste de contacts.
Former un groupe	Permet de constituer dynamiquement un groupe sur la base des liens sociaux.
Lister les liens	Permet d'afficher tous les liens sociaux d'un contact sélectionné.
Modifier un contact	Permet d'apporter des changements aux informations d'un contact.
Rechercher un contact	Permet de retrouver un contact à partir d'un critère.
Sélectionner un contact	Permet de choisir un contact afin d'effectuer des opérations telles que la suppression, la modification, ou l'affichage d'informations sur le contact.
Supprimer un contact	Permet d'effacer de la liste de contacts un contact sélectionné.

Trier les contacts	Permet de classer les contacts par ordre alphabétique croissant ou décroissant

III.2.3 Description de quelques cas d'utilisation

L'enchainement des opérations permettant de réaliser le cas d'utilisation « Enregistrer un contact » est présenté dans le tableau 5.

Tableau 5. Description textuelle du cas d'utilisation « Enregistrer un contact »

CU: « Enregistrer un contact ».	
Résumé : permet de sauvegarder un contact dans le répertoire téléphonique.	**Acteurs :** utilisateur **Version :** 1.0 **Date de création :** 07/02/2017 **Responsable :** Rashid ZONGO
Pré conditions :	
Scénario nominal : 1. L'utilisateur accède à l'onglet « enregistrer contact ». 2. Le système lui affiche le formulaire d'enregistrement de contact. 3. L'utilisateur renseigne les informations concernant le contact et valide. 4. Le système vérifie les champs remplis. (E) (A) 5. Le système sauvegarde le contact et le notifie à l'utilisateur.	
Enchainements alternatifs : **A1 :** L'enchainement commence au point 3 du scénario nominal : l'utilisateur souhaite tisser un lien entre le nouveau contact et un autre déjà existant. 5. L'utilisateur sélectionne le type de lien qu'il souhaite établir entre les deux contacts. 6. L'utilisateur choisit le contact existant dans la base et valide. L'enchainement continue au point 4 du scénario nominal.	
Enchainements d'exception : **E1 :** L'enchainement commence au point 3 du scénario nominal : un champ obligatoire n'est pas rempli. 7. Le système notifie à l'utilisateur l'impossibilité d'enregistrer le contact. 8. L'utilisateur complète les informations et valide. L'enchainement continue au point 4 du scénario nominal. **E2 :** L'enchainement commence au point 3 du scénario nominal : le genre du contact existant n'est pas en adéquation avec le type de relation sélectionnée. 5. Le système notifie à l'utilisateur l'impossibilité d'enregistrer le contact. 6. L'utilisateur sélectionne la bonne information. L'enchainement continue au point 4 du scénario nominal.	
Post conditions : un contact est enregistré dans le répertoire téléphonique.	

Le tableau 6 présente l'enchainement des opérations pour effectuer une recherche de contacts.

Tableau 6. Description du cas d'utilisation « Rechercher un contact »

CU: « Rechercher un contact ».	
Résumé : permet de retrouver un contact en fonction soit du numéro, soit du nom, soit d'un lien social.	**Acteurs :** utilisateur **Version :** 1.0 **Date de création :** 07/02/2017 **Responsable :** Ibraïma DAGNOGO
Pré condition :	
Scénario nominal : 1. L'utilisateur accède au champ de recherche. 2. L'utilisateur choisit le type de recherche. (A) 3. L'utilisateur saisit les données nécessaires et valides. 4. Le système lui affiche le résultat qui correspond au(x) contact(s) recherché(s).	
Enchainements alternatifs : **A1 :** L'utilisateur souhaite effectuer une recherche par lien. L'enchainement commence au point 2 du scénario nominal : 1. L'utilisateur sélectionne le type de lien qui existe entre le contact à chercher et un autre. 2. L'utilisateur choisit le contact existant dans la base et valide. L'enchainement continue au point 4 du scénario nominal. **A2 :** L'utilisateur souhaite effectuer une recherche par contact (nom ou numéro). L'enchainement commence au point 2 du scénario nominal : 3. L'utilisateur saisit le nom ou le numéro à rechercher et valide L'enchainement continue au point 4 du scénario nominal.	
Post condition : le contact recherché est affiché ou pas à l'utilisateur.	

III.2.4 Concepts du système

Les concepts généraux du système que nous retenons sont :

- un contact : c'est l'ensemble des coordonnées qui identifient une personne ;
- un groupe : c'est un ensemble constitué de contacts.

III.3 Modélisation du système

III.3.1 Diagramme de cas d'utilisation

La figure 5 présente le diagramme de cas d'utilisation définissant les exigences fonctionnelles attendues, les acteurs (utilisateurs du système) ainsi que les relations qui unissent acteurs et fonctionnalités.

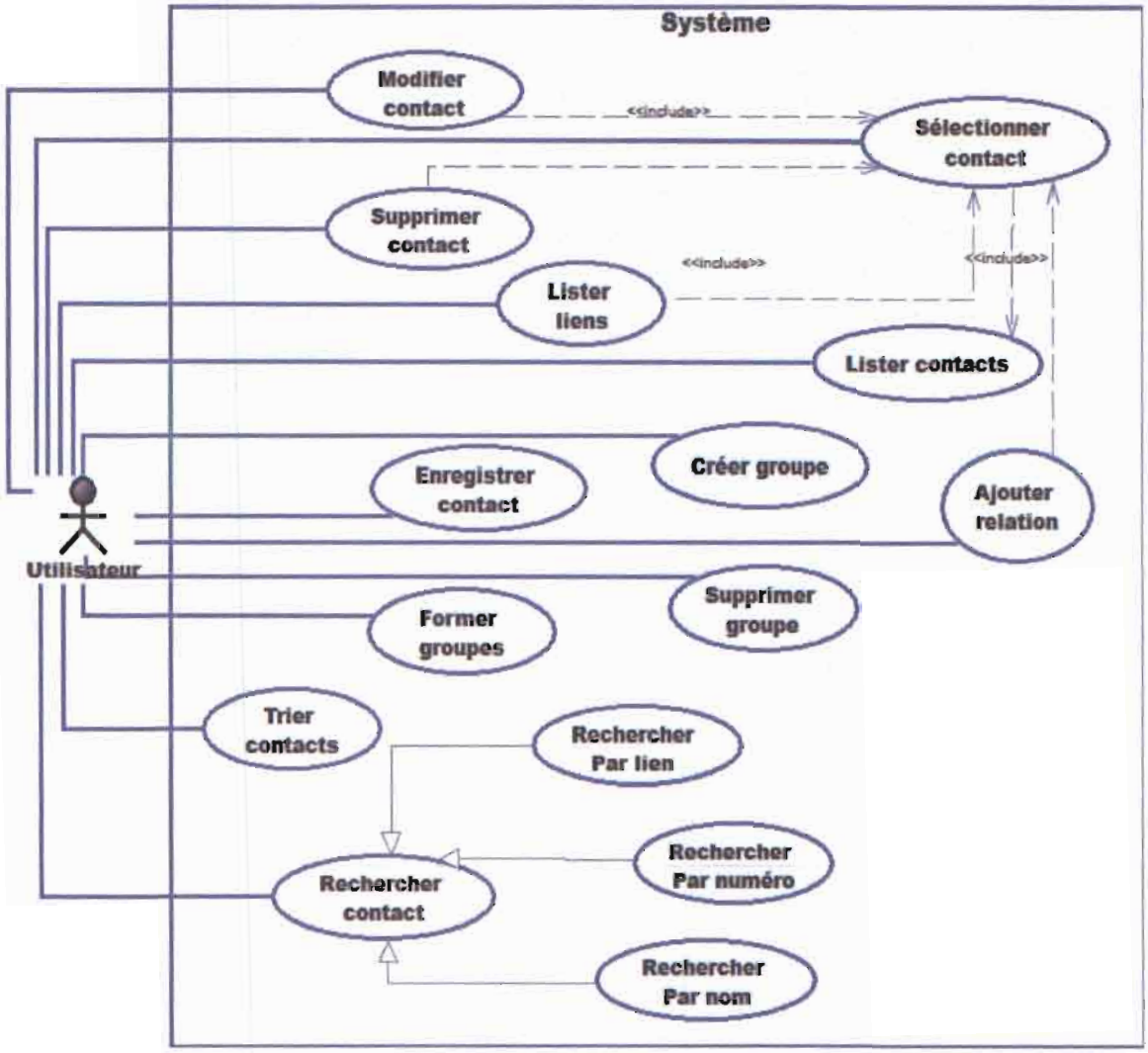


Figure 5. Diagramme de cas d'utilisation

III.3.2 Diagramme de séquences

Les diagrammes de séquences décrivent de manière plus formelle les interactions ou collaborations entre les acteurs (ou objets) dans le système. Ils permettent de représenter des collaborations entre objets selon un point de vue temporel, l'accent étant mis sur la chronologie des envois de messages.

La figure 6 présente la séquence des opérations permettant de réaliser le cas d'utilisation « Enregistrer un contact ».

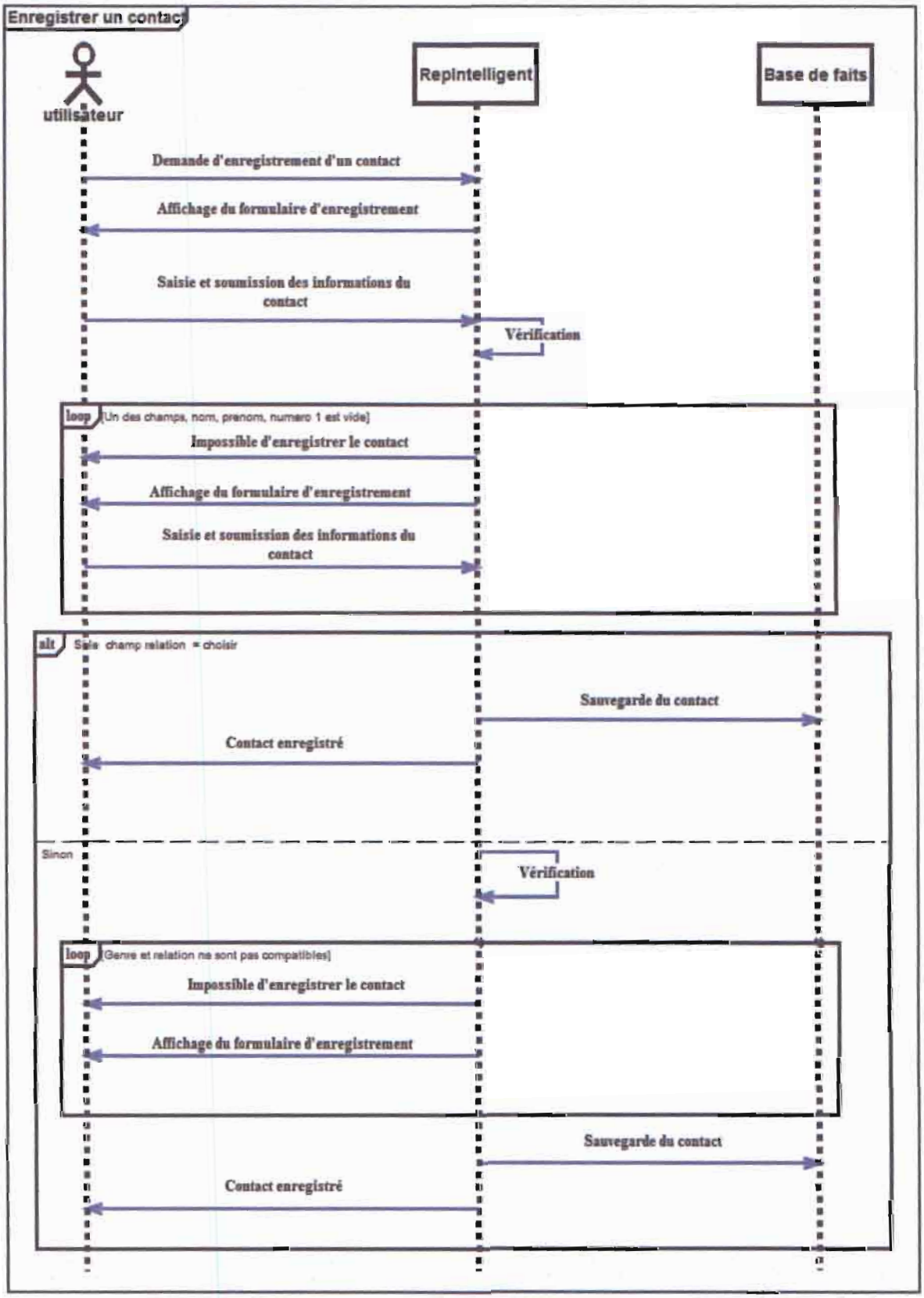


Figure 6. Diagramme de séquence du cas d'utilisation « Enregistrer un contact »

La figure 7 présente la séquence des opérations permettant de réaliser le cas d'utilisation « Rechercher un contact ».

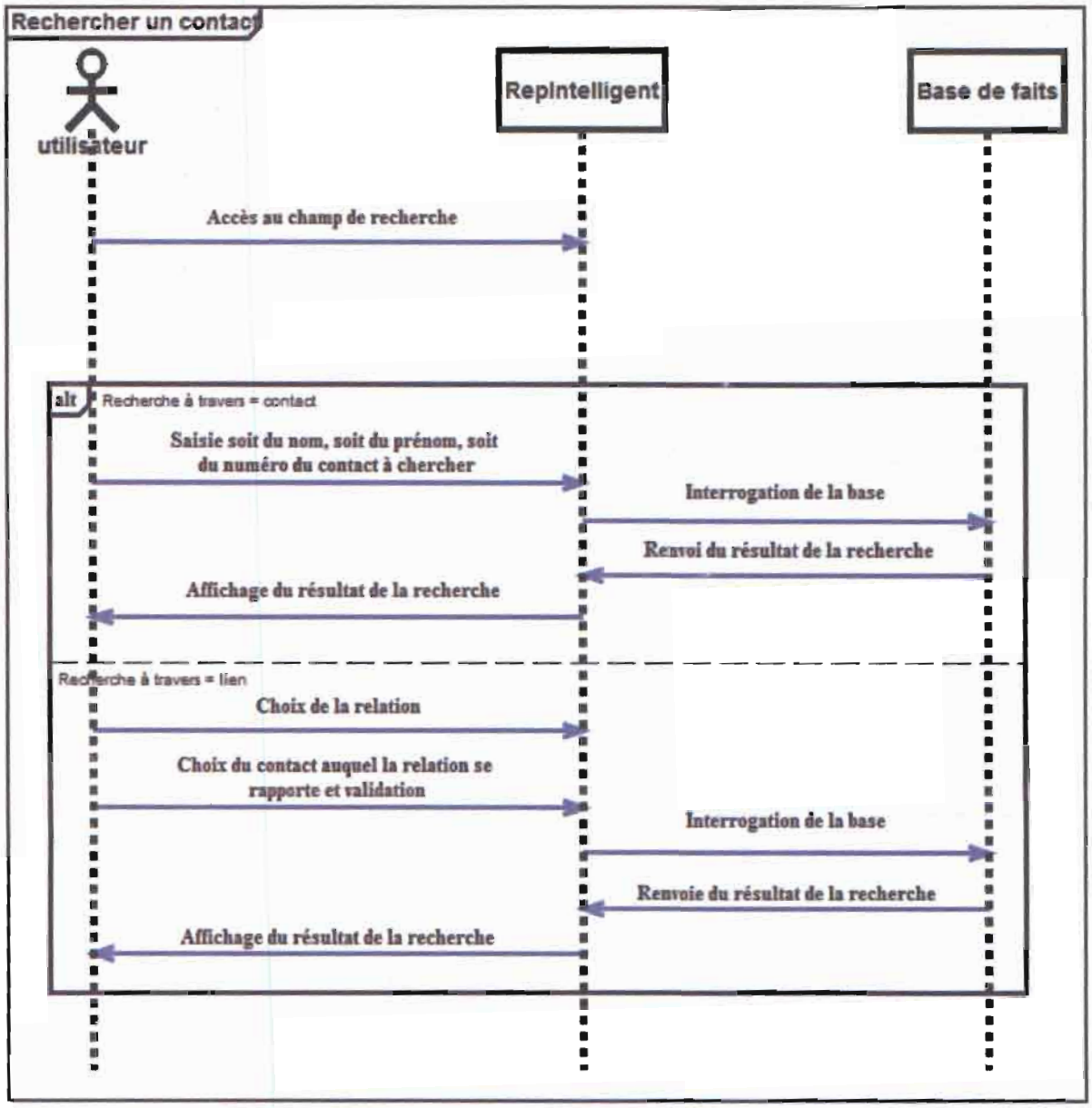


Figure 7. Diagramme de séquence du cas d'utilisation « Rechercher un contact »

III.3.3 Diagramme de classes

Il s'agit de la représentation en UML des différents concepts dans le cadre de notre projet. Cette modélisation donne une idée des attributs qui seront stockés. La figure 8 présente l'illustration du diagramme de classes.

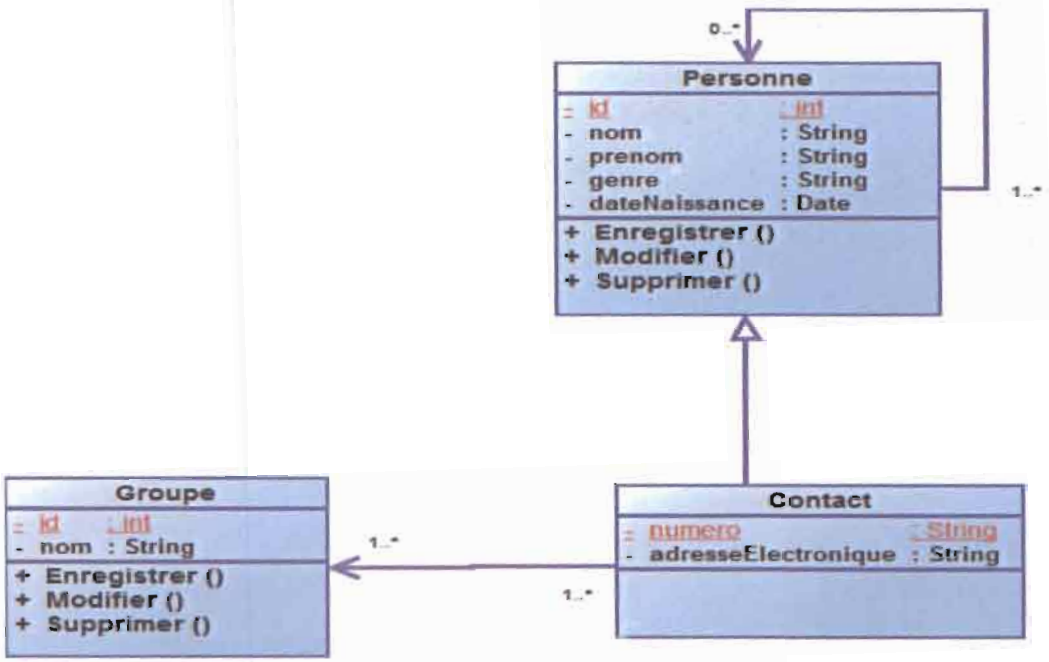


Figure 8. Diagramme de classes

N.B : ce diagramme de classes ne sera pas transformé en une base de données relationnelle.

III.4 Etude algorithmique et calcul de complexité

III.4.1 Définition

Un algorithme est une méthode permettant de résoudre efficacement un problème donné en un temps fini et cela quelles que soient les données à traiter. Il n'est pas à confondre avec un programme qui lui, décrit une méthode qui sera ensuite implémentée dans un langage de programmation.

III.4.2 Quelques algorithmes

Notre répertoire apporte de nouvelles fonctionnalités telles que :

- Recherche de liens : il arrive que nous retrouvions dans la liste de contacts, un contact dont la simple vue ne nous donne aucune information sur qui il est. La fonctionnalité permet d'afficher le contact en question ainsi que les autres contacts avec lesquels il a une relation sociale. En effet, elle permet d'avoir plus d'informations sur un contact en fonction de ses liens sociaux avec d'autres contacts existant dans le répertoire. L'algorithme suivant illustre cet aspect.

Variables
j : entier naturel ;
C_j : contact j donné ;
Début
Lire C_j ;
Afficher liens sociaux de C_j ;

Vérifier liens sociaux ;
Si C_j
Alors sélectionner C_j ;
FinSi
Fin

- Recherche par liens : cette fonctionnalité permet de rechercher un contact à travers un lien social en supposant qu'on ne se rappelle que de ce lien en ce qui concerne le contact.

Variables
j : entier naturel ;
C_j : contact j donné ;
Début
Lire C_j
Afficher liens sociaux de C_j ;
Eliminer liens sociaux ;
Vérifier liens sociaux ;
Si C_j
Alors sélectionner C_j ;
FinSi
Fin

- Résolution d'homonymie : permet de sélectionner un contact parmi plusieurs contacts du même nom. La fonction permet, parmi plusieurs contacts possédant le même nom, de choisir celui dont on a besoin en analysant les différentes relations qui s'affichent pour le contact. L'algorithme suivant met en œuvre cela.

Variables :
j, i, l : entier naturel ;
C_j : contact j donné ;
Début
Lire C_j ;
Pour i = 1 à l
Afficher liens sociaux de C_i ;
Vérifier liens sociaux de C_i ;
Si C_i = C_j
Alors sélectionner C_i ; sortir ;
Fin si
Fin pour
Fin

III.4.3 Calcul de complexité : complexité temporelle

III.4.3.1 Définition

La complexité temporelle d'un algorithme est une mesure de sa performance asymptotique dans le pire des cas. C'est une mesure qui se base sur des données très grandes (les petites données ne sont pas assez informatives) et s'intéresse également à la performance de l'algorithme dans

les situations ou le problème prend le plus de temps à être résolu. Cela permet de se rassurer que l'algorithme ne prendra jamais plus de temps que ce que l'on a estimé.

III.4.3.2 Notation $O(.)$

Les calculs à effectuer pour évaluer le temps d'exécution d'un algorithme peuvent parfois être longs et pénibles. De plus le degré de précision qu'ils requièrent est souvent inutile. On aura donc recours à une approximation de ce temps de calcul, représentée par O(.). L4es calculs respectent des règles (voir annexe).

Soit n la taille des données à traiter. On dit qu'une fonction f(n) est en O(g(n)) (en grand O de g(n)) s'il existe un seuil à partir duquel la fonction f (.) est toujours dominée par la fonction g(.), à une constante multiplicative fixée près.

III.4.3.3 Algorithme de recherche de liens

La complexité de l'algorithme de recherche de liens est illustrée dans le tableau 7 ci-dessous.

Tableau 7. Complexité : recherche de liens

Programme JAVA	**Nombre d'opérations**	**Complexité**

import java.util.Scanner;
 	
public class Main {
public static void main(String[] args){
Scanner sc = new Scanner(System.in);
 	
int n=N; //nombre total de contacts
int i = 0 ;
 	
String C_j = sc.nextLine (); // lecture du contact j
 	
While (i<n) {
 	
If(C_i) {// si le contact i est en relation avec le contact C_j
 	
System.out.println C_i; //sélection de Ci
}
i=i+1;
}
}
}
	Initialisation : 1
	Initialisation : 1
	
	Affectation : 1
	
	Itérations : au plus n
	
	Vérification : 1
	
	Renvoi : 1
	
	Addition + Affectation : 2
		O (1)
		O (1)
		
		O (1)
		
		O (n)
		
		O (1)
		
		O (1)
		
		O (1)

Nombre total d'opérations: $1 + 1 + 1 + n * ((1 + 1) + 2) = 3 + 4n$

La complexité de l'algorithme est donc $\boldsymbol{O}\ (\boldsymbol{3 + 4n}) = \boldsymbol{O}\ (\boldsymbol{n})$.

III.4.3.4 Algorithme de recherche par liens

Dans le tableau 8, nous calculons la complexité de l'algorithme lié à la recherche de contact par liens.

Tableau 8. Complexité : recherche par liens

Programme JAVA	Nombre d'opérations	Complexité

import java.util.Scanner;
 	
public class Main {
public static void main(String[] args){
Scanner sc = new Scanner(System.in);
 	
int n=N; //nombre total de contacts
int i = 0 ;
 	
String C_j = sc.nextLine (); // lecture du contact Cj
String Lien = sc.nextLine (); // lecture du lien
While (i<n) {
 	
If(C_i) {// si le contact i est en relation avec le contact j selon le lien défini
 	
System.out.println C_i; // sélection de C_i
}
i=i+1;
 	
}
}
}
	Initialisation : 1
	Initialisation : 1
	
	Affectation : 1
	Affectation : 1
	Itérations : au plus n
	
	Vérification : 1
	
	Renvoi : 1
	
	Addition + Affectation : 2
		O (1)
		O (1)
		
		O (1)
		O (1)
		O (n)
		
		O (1)
		
		O (1)
		
		O (1)

Nombre total d'opérations: $1 + 1 + 1 + 1 + n * ((1 + 1) + 2) = 4 + 4n$

La complexité de l'algorithme est donc $\boldsymbol{O}\ (\boldsymbol{4 + 4n}) = \boldsymbol{O}\ (\boldsymbol{n})$.

III.4.3.5 Algorithme de résolution d'homonymie

Le tableau 9 présente le calcul de complexité de l'algorithme permettant de résoudre l'homonymie.

Tableau 9. Complexité : Résolution d'homonymie

Programme JAVA	Nombre d'opérations	Complexité

import java.util.Scanner;
 	
public class Main {
public static void main(String[] args){
Scanner sc = new Scanner(System.in);
 	
int n=N; //nombre total de contacts
int m=H ; //nombre d'homonymes du contact C_j dans le répertoire
int i = 0 ;
int j=1 ;
 	
String C_j = sc.nextLine (); // lecture du contact C_j
While (j<=m) {
 	
While (i<n) {
 	
If(C_i) {// si le contact i est en relation avec le contact j
 	
System.out.println C_i; // sélection de C_i
}
i=i+1;
 	
} j=j+1;
}
}
}
	Initialisation : 1
	Initialisation : 1
	
	Initialisation : 1
	Initialisation : 1
	
	Affectation : 1
	Itérations : au plus m
	
	Itérations : au plus n
	
	Vérification : 1
	
	Renvoi : 1
	
	Addition + Affectation : 2
	
	Addition + Affectation : 2
		O (1)
		O (1)
		
		O (1)
		O (1)
		
		O (1)
		O (m)
		
		O (n)
		
		O (1)
		
		O (1)
		
		O (1)
		
		O (1)

Nombre total d'opérations: $1 + 1 + 1 + 1 + 1 + m * [(n * ((1 + 1) + 2)) + 2] = 5 + m(4n + 2)$

La complexité de l'algorithme est donc $\boldsymbol{O(5 + m(4n + 2)) = O(mn)}$.

Conclusion

Ce chapitre a permis de cerner le cadre de réalisation de ce projet. En effet la spécification des fonctionnalités attendues a permis de mieux comprendre les attentes. La phase de conception, elle, a permis de voir comment seront implémentées les différentes fonctionnalités, en décrivant de manière précise et détaillée, le fonctionnement du futur système afin d'en faciliter sa réalisation.

4

CHAPITRE IV : IMPLEMENTATION DU REPERTOIRE TELEPHONIQUE BASE SUR LES RELATIONS SOCIALES

Introduction

Ce chapitre aborde le travail à faire sous deux volets en se basant sur les choix technologiques. Il présente de façon détaillée le travail qui a été concrètement réalisé en présentant des illustrations. En fonction de la technologie, des modules de l'application sont présentés et aussi une partie est consacrée à la critique de chaque catégorie de technologies.

IV.1 Outils de développement

IV.1.1 Langage de développement

Un langage de développement est une notation conventionnelle destinée à produire des programmes informatiques. Il existe plusieurs types de langages de développement. Le tableau 10 illustre une comparaison entre les langages Java, C#.

Tableau 10. Etude comparative des langages de développement

Langage de développement	Avantages	Inconvénients
Java	– Portabilité des programmes; – Facilite le déploiement d'applications à base de composant ; – Intégration avec les systèmes d'information existants ; – Documentation automatique du code (Javadoc) ;5 – Sécurité	– Lenteur d'exécution des programmes due à l'interprétation ; – Gourmand en ressources à cause de la JVM.
C#	– Orienté objet ; – Meilleure si on n'utilise que des outils Microsoft ; – Outils client généralement très bien réalisés.	– Faiblesse de performance ; – Onéreux ; – Non portable, champ limité

Pour notre projet, nous avons opté pour le langage JAVA qui est en constante amélioration. Chaque nouvelle version apporte son lot de nouveautés au niveau de l'API et apporte de nettes améliorations au niveau des performances et de la stabilité. C'est un langage orienté objet très puissant et disposant de nombreuses librairies tierces. Il permet d'obtenir des applications bien

structurées, modulables et maintenables.

IV.1.2 Environnement de développement

Un EDI (Environnement de Développement Intégré) ou en anglais IDE (Integrated Development Environment) est un logiciel regroupant un ensemble d'outils facilitant le développement des applications. Nous distinguons plusieurs EDI tels que Netbeans, Eclipse, Visual studio… Dans le tableau 11 présente une comparaison des EDI.

Tableau 11. Etude comparative de quelques EDI

EDI	Avantages	Inconvénients
Netbeans	– Fonctionnalité de partage de projet ; – Débogage complet ; – Support de test et de gestion de source très performant ; – Facile et simple d'utilisation ; – Licence GPL V2 et gratuit ;	– Nécessite le JDK ; – Outil de correction d'orthographe moyen ;
Eclipse	– Débogage complet ; – Support de test performant ; – Plus large bibliothèque de plugin ; – Licence GPL V2 et gratuit ;	– Prise en main compliqué ;
Visual studio	– Simple d'utilisation ; – Conviviale, rapide et efficace ; – Support de qualité ;	– Très couteux pour la version professionnelle ; – Ne supporte pas Java ; – Outil propriétaire;

Dans le cadre de ce projet, nous avons choisi l'EDI Netbeans 8.0.2. En effet, Netbeans est un EDI qui s'adapte aux langages de programmations (Java, JavaScript, Python, PHP, Groovy, C/C++, etc.), aux ressources et aux outils disponibles sur le système. Il est ainsi possible de développer facilement des applications. De plus sa licence est gratuite.

IV.1.3 Construction d'ontologies avec Protégé 2000

IV.1.3.1 Composantes d'une ontologie

Une ontologie est construite autour de plusieurs notions telles que :

– les concepts ou classes : ils représentent un ensemble d'objets et leurs propriétés communes ;

- les propriétés : il existe deux types de propriétés dont les attributs sont des propriétés simples qui se rapportent à une instance et les rôles qui expriment l'association des concepts ;
- les instances : il s'agit des individus liés à un concept ;
- les relations : il s'agit de relations binaires entre un concept général et un concept plus spécifique. Elles induisent une hiérarchie de spécialisation (une taxinomie). Il en est de même que chez les rôles ;
- les axiomes, les règles et les restrictions : ils expriment ce qui ne peut l'être comme concept ou propriété.

Pour la création de l'ontologie, nous ne tiendrons pas compte des instances car celle-ci devront être insérées à partir de l'application avec laquelle l'ontologie interagira. Les restrictions et axiomes seront définis lors de la création et la définition des classes et des propriétés. Les règles seront écrites dans un fichier texte et seront gérées par le Framework JENA.

IV.1.3.2 Classes de l'ontologie

La première étape, lors de la création d'une ontologie OWL, est d'organiser les différents éléments nécessaires en classes. La hiérarchie des classes est extrêmement importante car, à l'instar de la programmation orientée objet, les classes filles hériteront des propriétés et caractéristiques des classes parentes [10].

Protégé 2000 offre une interface de gestion des classes. A partir de cette interface, on crée les classes en suivant une hiérarchie qu'on aura choisie. Aussi, pour chaque classe, il est important de la définir, de lui donner une sémantique.

La figure 9 ci-après montre les classes constitutives de notre ontologie ainsi que leur hiérarchie. Cette ontologie se rapporte plus à une famille de personnes.

Nous avons deux concepts mère : Personne et Genre. Le concept « Genre » possède deux instances : masculin et féminin. Le concept « Personne » possède des sous-classes dont deux sous-classes à partir desquelles toutes les autres sous-classes sont définies : la sous-classe « homme » et la sous-classe « femme ».

Un homme est défini comme étant une personne ayant pour sexe masculin et une femme comme une personne ayant pour sexe féminin. Ainsi donc une personne est soit un homme, soit une femme. Toutes les autres classes sont définies à partir des classes homme, femme et Personne.

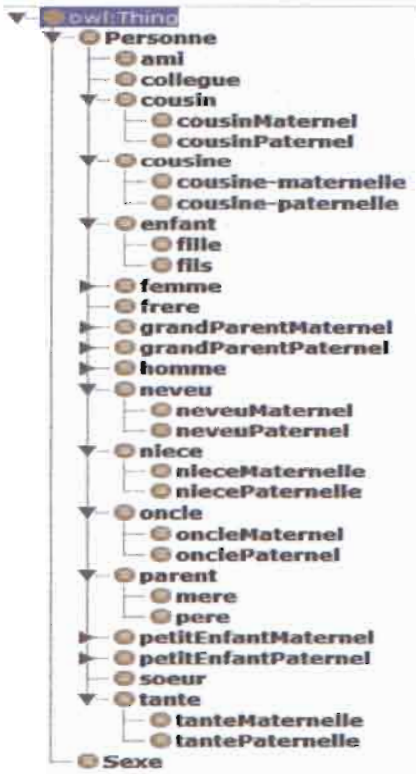


Figure 9. Vue partielle de la hiérarchie des concepts

IV.1.3.3 Propriétés

En OWL sous Protégé 2000, il y a deux types de propriétés qui sont : les « *Object Properties* » et les « *Data Properties* ». Il existe un troisième type de propriétés « *Annotations Properties* ». Les « *Object Properties* » et les « *Data Properties* » peuvent être notées comme des propriétés d'annotation.

Une notion importante pour la gestion des propriétés est la définition pour chacune d'elles de leur domaine et de leur portée. Le domaine permet de déterminer à quelles classes de sujets une propriété peut être ajoutée. La portée détermine à quelles classes l'objet doit appartenir [10]. Concernant les propriétés d'objet, l'interface de gestion se présente comme sur la figure 10. Cette figure présente également la liste des propriétés d'objet créées pour les besoins du projet.

Chaque propriété d'objet peut être :

- fonctionnelle : elle définit que l'ensemble des objets liés à un sujet donné par une propriété de ce type sont égaux ;
- inverse fonctionnelle : une propriété est inverse fonctionnelle si son inverse est une propriété fonctionnelle ;
- symétrique : elle définit que si un sujet A est lié à un objet B par la propriété P alors B est lié à A par la propriété P ;

- asymétrique : elle définit que si un sujet A est lié à un objet B par la propriété P alors B ne peut en aucun cas être lié à A par la propriété P ;
- transitive : définit que si un sujet A est lié à un objet B par la propriété P et B est lié à C par la propriété P alors A est lié à C par la propriété P ;
- réflexive : une propriété P est dite réflexive si elle se rapporte à l'individu A lui-même.
- inverse réflexive : c'est l'inverse de la réflexivité [10].

De plus la propriété « inverseOf » permet de définir la propriété inverse à une propriété donnée. Pour chaque propriété créée, il faut préciser le domaine et l'intervalle de chacune. La hiérarchie également est très importante comme au niveau des classes.

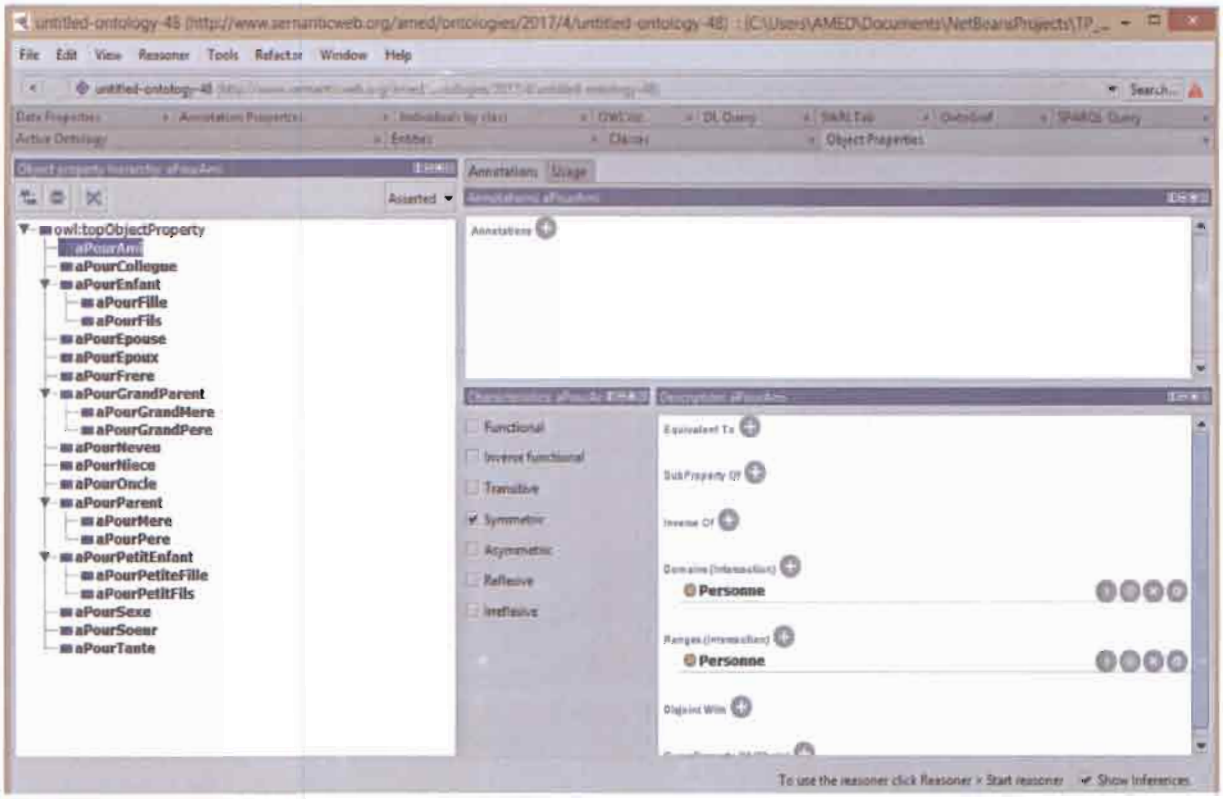


Figure 10. Interface de gestion des propriétés d'objets

Les « *data properties* » sont les équivalents des attributs d'une classe dans la programmation orientée objet. La figure 11 ci-dessous présente les « *data properties* » concernant notre ontologie. Toutes les propriétés de données ont pour domaine « Personne » et pour intervalle « String » sauf la propriété « identifiant » dont l'intervalle est un « Integer ». Ces propriétés sont communes à chaque instance de l'ontologie. Chaque instance de notre ontologie aura un identifiant, un nom, un prénom et la possibilité d'avoir trois numéros.

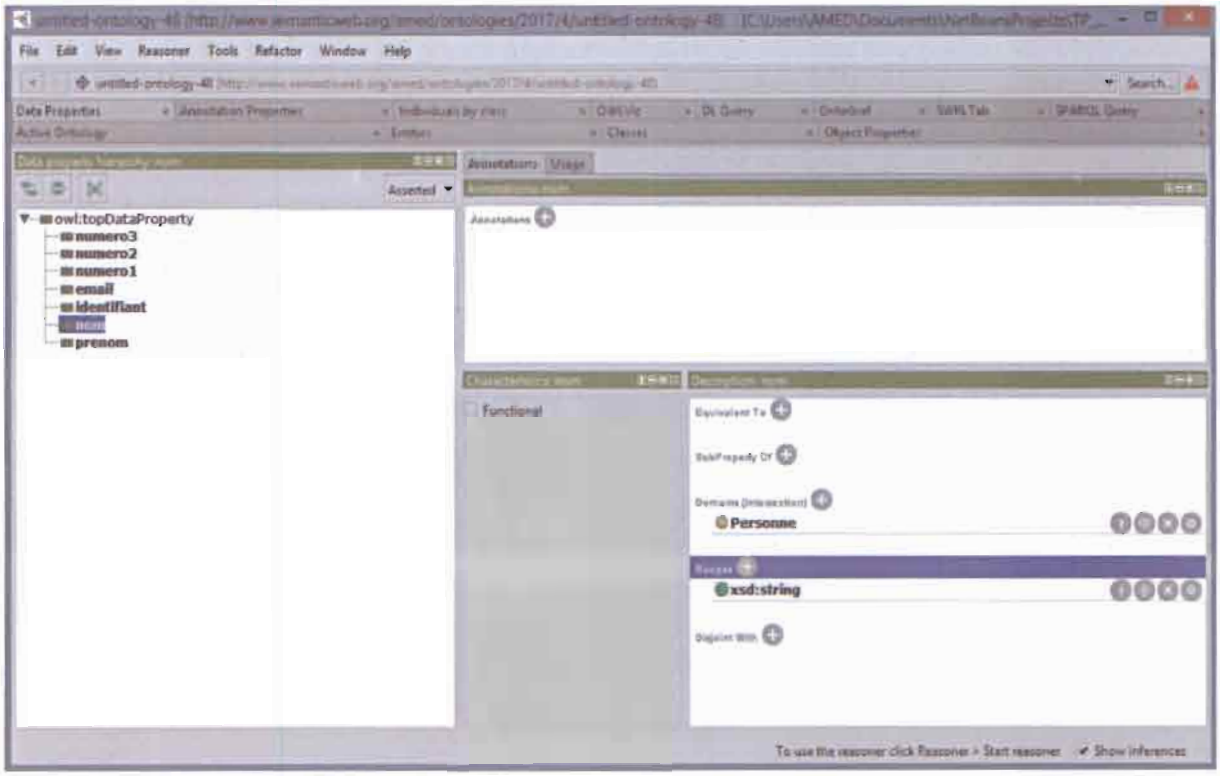


Figure 11. Interface de gestion des propriétés de données

IV.1.4 Architecture du système

IV.1.4.1 Architecture technique des composantes

L'architecture décrit d'une manière symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions. Elle explicite les différentes composantes qui vont interagir pour former le système à mettre en place.

IV.1.4.1.1 Technologies de l'intelligence artificielle classique

La figure 12 présente une ébauche d'architecture de notre système.

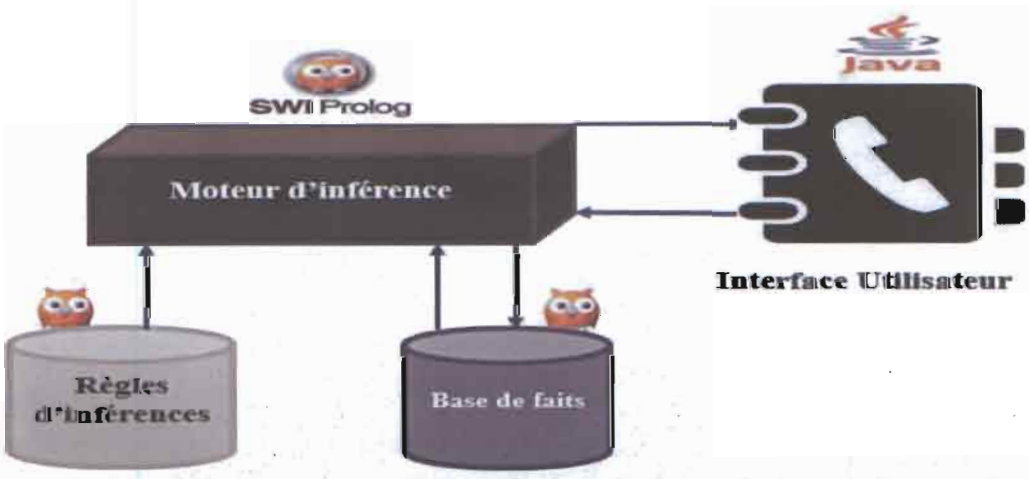


Figure 12. Architecture du système : approche basée sur Prolog

Cette architecture communique avec sa base de faits qui est constituée de fichiers Prolog, par l'intermédiaire d'un moteur d'inférences, SWI-Prolog. Ce moteur d'inférences utilise une base de règles qui est aussi un fichier Prolog. La communication entre la base de faits et le moteur d'inférences est bidirectionnelle, l'une pour l'extraction des informations et l'autre pour leur stockage. Le répertoire peut donc envoyer des requêtes à la base de faits ou y stocker des contacts. Les règles servent à l'établissement des relations qui y sont clairement définies.

IV.1.4.1.2 Technologies des donnée liées

L'architecture de cette application ne diffère pas beaucoup de l'architecture de celle utilisant la base Prolog. L'application se compose :

- d'une base ontologique servant de base de données ;
- d'un moteur d'inférences Jena ;
- d'une base de règles écrite avec le langage de règles du moteur d'inférences Jena.

La figure 13 montre l'architecture de l'application à base d'ontologie.

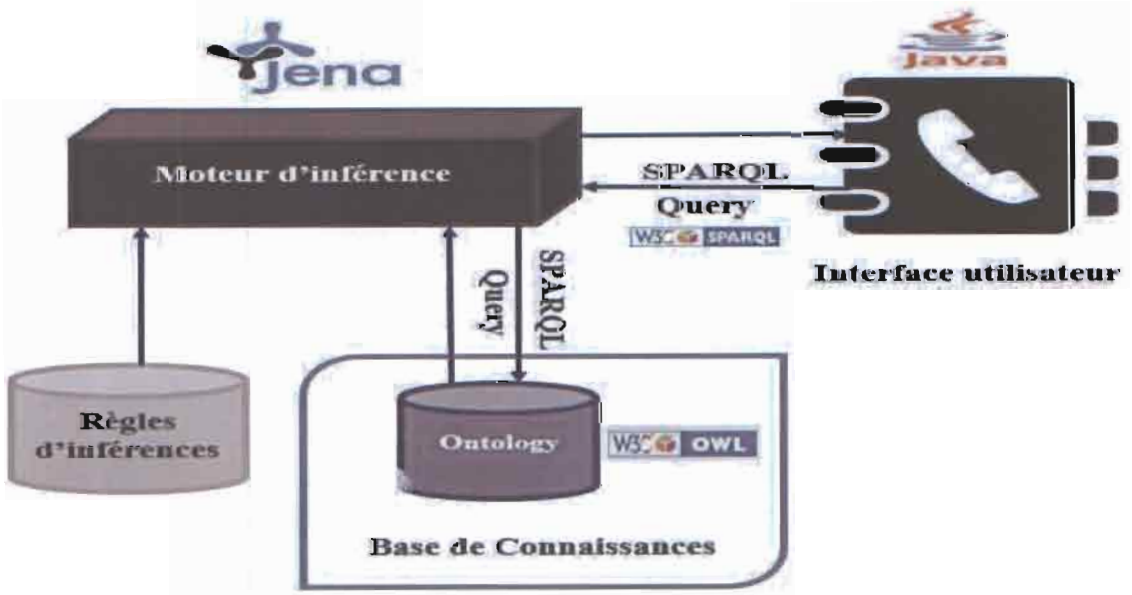


Figure 13. Architecture du système : approche à base ontologique

IV.1.4.2 Architecture physique des données

IV.1.4.2.1 Technologies de l'intelligence artificielle classique

Une base de faits est l'une des entrées d'un moteur d'inférences. C'est un ensemble de connaissances appelées "faits" et considérés comme vrais. À partir de ces faits, le moteur d'inférences va leur appliquer les règles issues de sa base de règles pour en déduire d'autres faits;

et ainsi résoudre un problème de logique. En effet notre base de faits est construite à partir des enregistrements de contacts que l'utilisateur effectue. Ainsi la base de faits est constituée des différents fichiers Prolog créés. Nous avons dix-huit fichiers Prolog. Chacun de ces fichiers contient des informations bien précises suivant un schéma bien défini. Aussi il existe un lien entre tous les fichiers. De façon globale, la figure 14 ci-après présente une illustration des liens entre les fichiers les plus représentatifs.

Nous avons également un fichier de règles qui constitue notre base de règles. Ce fichier contient l'ensemble des règles sur lesquelles le système se réfère pour faire des inférences afin de répondre aux requêtes qui lui sont transmises. Ci-dessous, quelques règles utilisées par notre application :

pere(X,Y) :- parent(X,Y),homme(X).
mere(X,Y) :- parent(X,Y),femme(X).
fils(X,Y) :- enfant(X,Y),homme(X).
fille(X,Y) :- enfant(X,Y),femme(X).
relation(X,Y):- fille(X,Y).
relation(X,Y):- frere(X,Y).

En effet la règle « *fils(X,Y) :- enfant(X,Y), homme(X).* » se lit comme suit : X est le fils de Y si X est un enfant de Y et X est un homme. Aussi la règle « *relation(X,Y) :- frere (X,Y).* » se lit X est en relation avec Y si X est le frère de Y. Dans notre projet, on dira que deux personnes sont en relation si elles sont liées par une quelconque relation sociale.

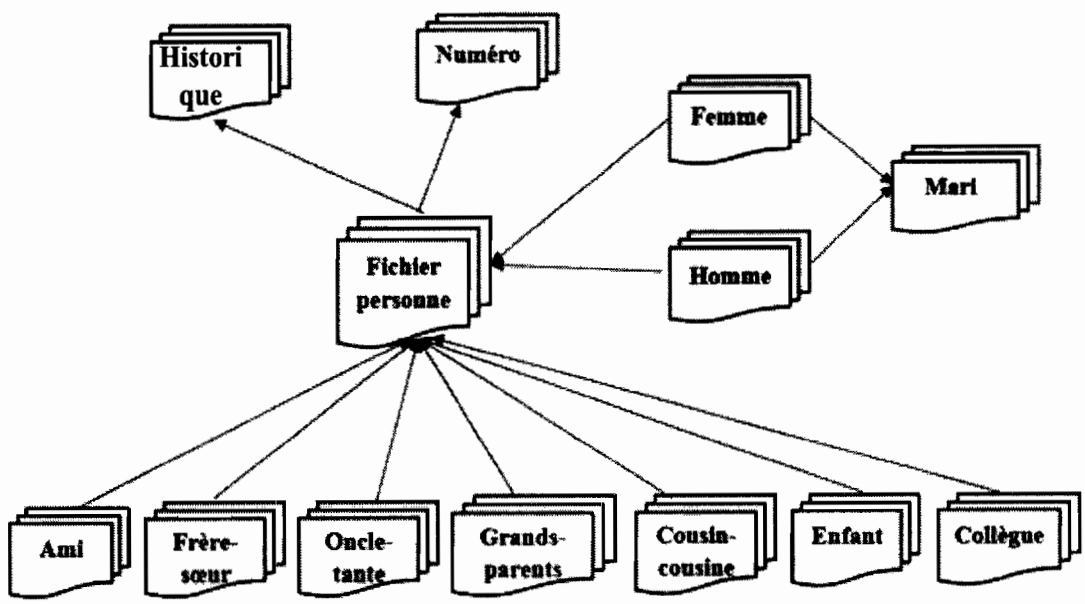


Figure 14. Implantation physique des contacts

IV.1.4.2.2 Technologies des données liées : ontographe

L'ontographe représente la structure physique de notre ontologie. Il explicite les différents liens entre les concepts de l'ontologie. La représentation de la figure 15 n'explicite que des liens de sous classes entre les concepts. Elle donne une idée sur la façon dont est structurée notre base de connaissances.

A côté de l'ontologie, nous avons un fichier de règles écrites dans le langage de règles de l'API Jena. Ces règles doivent servir à l'inférence, à la déduction de nouvelles connaissances et aussi à la définition de certaines relations entre deux contacts. Le fichier de règles respecte la structure de l'ontologie. Quelques règles de notre fichier :

[rule1: (?A rdf:type ns:homme)->(?A rdf:type ns:Personne)]

[rule2: (?A rdf:type ns:femme)->(?A rdf:type ns:Personne)]

Les deux premières traduisent qu'une personne est soit un homme, soit une femme.

[Fils2: (?A rdf:type ns:Personne) (?A ns:aPourParent ?B) (?A rdf:type ns:homme)->(?B ns:aPourFils ?A)]

La règle « *Fils2* » dit que si la personne A a pour parent une autre personne B et que A est un homme alors la personne B a pour fils la personne A.

[Fille1: (?A rdf:type ns:Personne) (?A ns:aPourEnfant ?B) (?B rdf:type ns:femme)->(?A ns:aPourFille ?B)].

La règle « *Fille1* » dit que si la personne A a pour enfant une autre personne B et que B est une femme alors la personne A a pour fille la personne B.

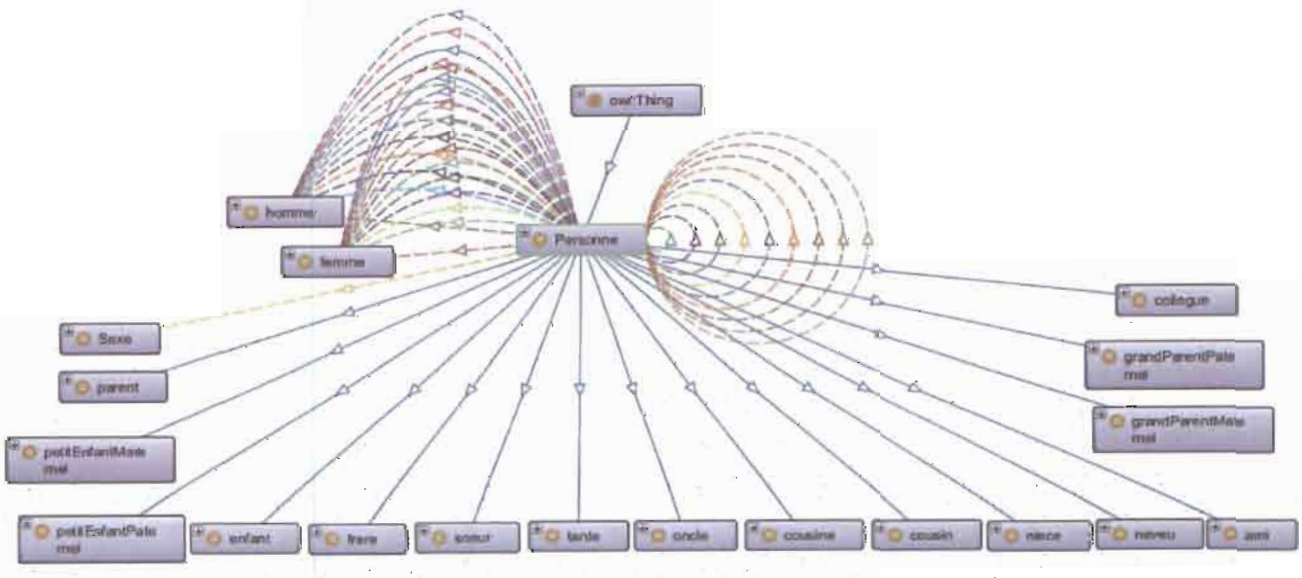


Figure 15. Vue partielle ontographe représentant les concepts du système

IV.1.5 Comparaison entre les différentes technologies utilisées

Dans le tableau 12 ci-après, nous présentons les forces, les faiblesses des différentes technologies utilisées dans la gestion des données.

A l'issu de cette comparaison, les technologies des données liées sont plus souples, plus facile à exploiter que les technologies de l'intelligence artificielle classique. Elles sont plus adaptées pour le système de gestion de données qui sera mis en place.

Tableau 12. Comparaison entre technologies classiques et technologies des données liées

Critères	Technologies classiques	Technologies des données liées
Forces et Faiblesses	+ Gestion efficace des bases de faits ; + Déduction de nouvelles connaissances à partir des données existantes ; + Dispose de nombreuses commandes pour la manipulation des données ; + Beaucoup de souplesse au niveau de l'édition des règles ; – Occupation importante de l'espace de stockage car trop de fichiers pour les bases de faits ; – Structure rigide ; – Manque de langages de requêtes ; – Les insertions faites à partir des commandes ne sont pas persistantes.	+ Gestion efficace des bases de connaissances ; + Déduction de nouvelles connaissances à partir des données existantes ; + Utilisation du langage de requêtes SPARQL pour la manipulation des données ; + Utilisation de peu de fonctions dans le programme pour la gestion des enregistrements et les recherches… + Détermination avec précision des types de relation que les contacts entretiennent entre eux ; + Stockage des relations de bases uniquement ; + Tissage efficace des relations ; + Très peu de fichiers utilisés (02 au total) ; – Occupation significative de la mémoire lors de l'exécution car trop d'inférences à faire.

Légende : + : Forces. – : Faiblesses.

IV.1.5.1 Réalisation : OntoRep

Cette partie présente un aperçu de quelques modules développés à travers quelques captures d'écran de l'application utilisant la base ontologique dénommée « OntoRep ». L'application fonctionne uniquement sur ordinateur.

IV.1.5.1.1 Interface d'accueil

Dans la version de l'application utilisant l'ontologie, l'interface d'accueil a subi une légère modification. Pour accéder à la liste de contacts, il suffit de cliquer sur l'icône de répertoire. En accédant à la liste de contacts, on a accès à toutes les autres fonctionnalités du répertoire. La figure 16 présente l'interface d'accueil et la liste de contacts.

Figure 16. Interface d'accueil (A) et liste de contacts (B)

IV.1.5.1.2 Profils de contacts

Comme dans l'application avec Prolog présentée plus haut, ici également nous avons deux profils : profil homme et profil femme ; dans les deux profils, un tableau est réservé pour les relations que le contact entretient avec d'autres contacts du répertoire. La figure 17 présente les deux profils.

Figure 17. Profil homme (A) et profil femme (B)

Pour voir la nature de la relation entre le contact et un autre contact dans le tableau relation sociale, il suffit de cliquer sur le contact en question. Les relations sont bien définies dans cette application et la majorité sont des déductions non enregistrées. La figure 18 illustre cela.

Figure 18. Illustration du lien existant entre deux contacts : ganama haoua est la mère de zongo farida (A) et ouedraogo djeneba est la grand-mère de zongo farida (B)

IV.1.5.1.3 Formulaire d'enregistrement de contacts

Concernant le formulaire d'enregistrement, l'utilisateur ne peut renseigner qu'un seul numéro. Il offre également la possibilité d'enregistrer un contact sans tissage de relation et avec tissage de relation. Il est également possible d'ajouter la date de naissance d'un contact. Le formulaire d'enregistrement est présenté dans la figure 19 ci-dessous.

Le bouton « Propriété » permet d'avoir plus d'informations sur le contact avec lequel on souhaite tisser une relation lors de l'enregistrement.

Figure 19. Formulaire d'enregistrement de contacts : formulaire vide (A) et exemple de formulaire rempli (B)

IV.1.5.1.4 Modification de contacts

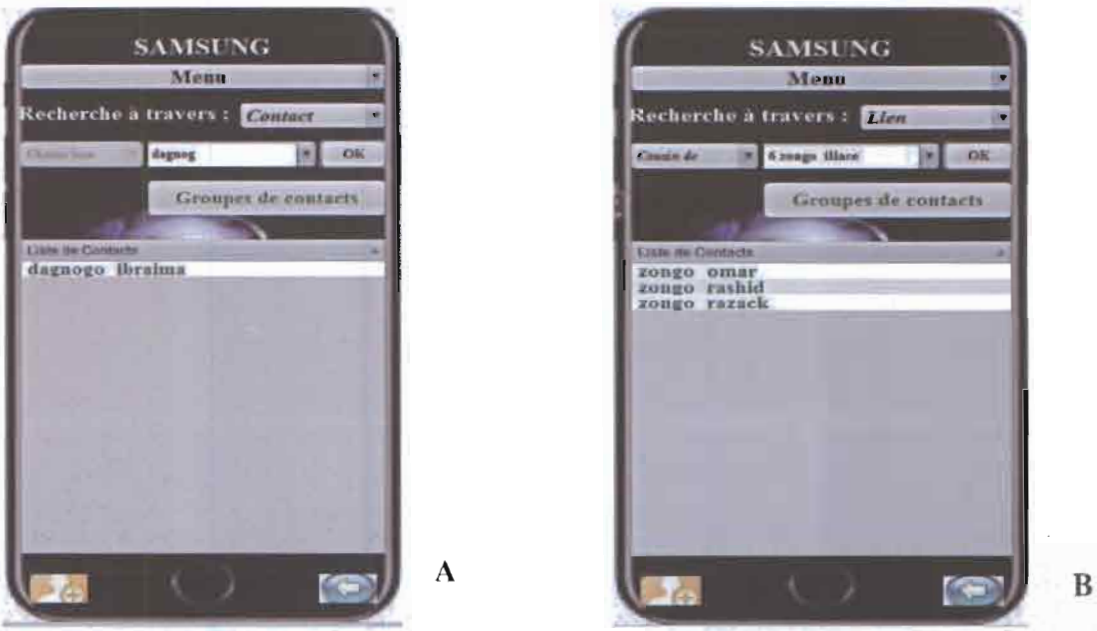
Dans cette application, la modification permet à l'utilisateur de mettre à jour un contact. Dans cet onglet, il a la possibilité d'ajouter d'autres numéros (02) au contact qu'il souhaite modifier. Le champ « date de naissance » est aussi modifiable. Pour accéder à la modification d'un contact, il suffit d'ouvrir son profil et cliquer sur « Modifier ». La figure 20 ci-après présente les champs modifiables.

A B C

Figure 20. Modification d'un contact : contact à modifier (A), champs à modifier (B) et contact modifié (C)

IV.1.5.1.5 Recherche de contacts

La recherche de contacts dans l'application à base ontologique s'effectue de la même manière qu'au niveau de l'application avec Prolog. Il y a deux types de recherche : l'une par lien et l'autre par numéro ou nom. Par exemple, nous allons rechercher dans la liste de contacts le contact avec le nom « dagnogo ibraima » et tous les cousins du contact « Zongo Iliace ». La recherche de contact est illustrée à la figure 21 ci-après.



A B

Figure 21. Recherche de contacts par nom (A) et recherche de contacts par lien (B)

Conclusion

Ce chapitre a fait étalage du travail concrètement réalisé à travers l'étude des technologies. A la suite de cette étude, un choix des technologies a été fait. Notre choix a porté sur les technologies des données liées pour la gestion des données. Des modules de l'application à base d'ontologie sont présentés également dans ce chapitre. Des fonctionnalités spécifiques seront ajoutées à l'application.

5

CHAPITRE V : VERS UN REPERTOIRE TELEPHONIQUE INTELLIGENT

Introduction

Ce chapitre a pour but de présenter les nouvelles fonctionnalités de notre répertoire qui lui confèrent l'aspect intelligent.

V.1 Fonctionnalités intelligentes

Les fonctionnalités intelligentes sont de deux types : la communication de groupes et le système de recommandations.

V.1.1 Communication de groupes

V.1.1.1 Description de la fonctionnalité

La communication de groupes part de l'utilisateur de l'application vers une catégorie d'individus bien définie dans le répertoire. En effet, la communication de groupes consiste à sélectionner une catégorie de personnes bien précise en se basant sur les relations qui existent entre l'utilisateur et cette catégorie de contacts. La constitution de groupes a donc pour critères principaux les relations entre les contacts dans le répertoire.

L'objectif est donc de pouvoir former des groupes de contacts selon les relations lorsqu'il y a lieu de partager une information (par exemple : un évènement de portée familiale telle qu'un mariage, de portée plus grande telle que des funérailles, ...) avec une certaine catégorie de personnes présentes dans le répertoire téléphonique.

Il est important de noter que les groupes formés sont constitués dynamiquement, pour un besoin donné et ne sont donc pas sauvegardés.

V.1.1.2 Présentation de captures d'écran

V.1.1.2.1 Accueil de la gestion de groupes

Pour accéder à l'interface de gestion de la communication de groupes, il faut à partir de l'interface affichant la liste des contacts cliquer sur « groupe de contacts ». La figure 22 présente l'interface de gestion de la communication de groupes.

Figure 22. Interface de gestion de la communication de groupes

V.1.1.2.2 Constitution de groupes

Pour former un groupe, à partir de l'interface de gestion de la communication de groupes, il faut sélectionner un contact à partir duquel il existe une relation par le biais de laquelle on peut retrouver tous les autres contacts qui doivent faire partie du groupe qu'on souhaite former. Par exemple, lorsqu'on souhaite former un groupe avec tous ses frères et sœurs, il suffit de sélectionner, à partir du bouton déroulant un de ses frères, et à partir de ce dernier, cocher les cases « frères » et « sœurs » et cliquer sur « valider ». Formons donc un groupe de tous les frères et sœurs de l'utilisateur sachant qu'un de ses frères est le contact Zongo Rashid. La figure 23 illustre notre exemple.

Figure 23. Critères de choix (A) et membres de groupe (B)

En cochant par exemple « tous les liens » uniquement, le groupe formé sera constitué de tous les contacts avec lesquels le contact Zongo Rashid est en relation. Cela est illustré par la figure 24.

Figure 24. Constitution de groupes avec le critère « tous les liens »

En utilisant seulement le critère « Tous », le groupe formé sera constitué de l'ensemble des contacts du répertoire.

Figure 25. Définition de la date de naissance

Figure 26. Message de suggestion ou de rappel

Conclusion

Ce chapitre a permis de présenter les fonctionnalités intelligentes en plus de la gestion des relations dont est dotée notre application. Ces fonctionnalités se résument en deux : la communication de groupes et le système de recommandations.

V.1.2 Système de recommandations

V.1.2.1 Description de la fonctionnalité

Le système de recommandations a pour but de présenter à l'utilisateur les contacts qui sont susceptibles de l'intéresser. Il s'agit donc d'un système permettant de faire des suggestions de contacts à l'utilisateur en fonction d'un certain nombre de critères. Les suggestions peuvent se baser sur les nombres d'appels effectués entre l'utilisateur et un contact. Le système de recommandations peut être vu aussi comme un système permettant de faire un rappel à l'utilisateur concernant un évènement sur un contact.

Cette deuxième vision de la recommandation est celle développée à ce jour dans notre application. En ce sens, notre application rappelle à l'utilisateur la date d'anniversaire d'un contact qu'il a enregistré et lui suggère de l'appeler ou de lui envoyer un message lui souhaitant un joyeux anniversaire.

Les systèmes de recommandations basés sur l'historique des appels sont toujours en cours d'études.

V.1.2.2 Présentation de captures d'écran

La date d'anniversaire se définit à la phase d'enregistrement d'un contact. L'application prendra en compte le jour et le mois renseignés pour les comparer à la date du système. La date du système devra être donc toujours la bonne. La figure 25 ci-après montre la manière dont la date de naissance est renseignée lors de l'enregistrement.

En enregistrant le contact « Dagnogo ibraima » avec la date de naissance 22/06/2003, à chaque 22/06, un message de suggestion sera envoyé à l'utilisateur pour lui rappeler l'anniversaire du contact. Ce message sera visible sur l'écran d'accueil comme le montre la figure 26 ci-après.

En appuyant sur le bouton « OK! », le message cessera d'apparaitre ; la même action sur « Quitter! », le message cessera d'apparaitre un moment puis recommencera à s'afficher.

Il est aussi important de noter que lorsque plusieurs contacts ont leur anniversaire le même jour, la même suggestion sera faite pour chacun d'eux.

CONCLUSION GENERALE

A travers ce travail, nous avons proposé une implémentation d'une nouvelle vision de liste de contacts des téléphones mobiles. Cette nouvelle vision du répertoire téléphonique, conçue et introduite par une équipe d'enseignants chercheurs de l'Ecole Supérieure d'Informatique, est basée sur les relations sociales. Elle consiste en un ensemble de contacts liés les uns aux autres par des liens sociaux tels que les liens familiaux, amicaux, etc. A travers ces liens sociaux, il est possible de gérer de manière très efficace le problème de l'homonymie. Deux versions du répertoire téléphonique correspondant ont été développées en JAVA : l'une est basée sur des technologies classiques de l'intelligence artificielle à savoir le langage spécialisé Prolog et l'autre est basée sur des technologies émergentes mettant en œuvre les ontologies. Chaque version présente des forces significatives mais aussi des faiblesses. L'application à base d'ontologie reste la plus stable au vu de l'unicité et de la structure de sa base de données. Sur cette version de l'application, en plus de la gestion efficace des relations sociales, deux fonctionnalités intelligentes ont été ajoutées : la communication de groupes et un système de recommandations. Certains aspects du système de recommandations sont toujours en phase d'études.

La finalité de l'application est d'être utilisée sur les smartphones fonctionnant avec le système Android. Ainsi donc, l'évolution vers une application mobile Android est à envisager.

REFERENCES BIBLIOGRAPHIQUES

[1] Min, Jun-Ki, Kim, Hee-Taek, et Cho, Sung-Bae. Social and personal context modeling for contact list recommendation on mobile device. In : *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 03*. IEEE Computer Society, 2008. p. 381-384.

[2] Shevade Sundaram, H. Lexing Xie. Exploiting Personal And Social Network Context For Event Annotation. In : *Multimedia and Expo.* IEEE International Conference. Beijing: 2007. p. 835 – 838.

[3] Gao, Ya, Zhang, Chunhong, Wang, Yi, *et al.* A directed recommendation algorithm for user requests based on social networks. In : *Embedded and Ubiquitous Computing (EUC), 2011 IFIP 9th International Conference on*. IEEE, 2011. p. 457-462.

[4] Lee, HyungJik, Park, JiEun, Ko, EunJung, *et al.* An agent-based context-aware system on handheld computers. In : *Consumer Electronics, 2006. ICCE'06. 2006 Digest of Technical Papers. International Conference on*. IEEE, 2006. p. 229-230.

[5] Pasteur, Poda, Anasthasie, Joëlle, Compaoré, Borlli, Michel, Jonas, Somé. Redesigning Mobile Phone Contact List to Integrate African Social Practices. In: *8th International Conference on e-Infrastructure and e-Services for Developing Countries*. 2016.

[6] Ukpe, Emmanuel et Mustapha, Smfd Syed. An Analysis of Six Standard Ontology Editing Tools for Capturing Entire Crop Processing. *International Journal of Computer Science and Information Security*, 2016, vol. 14, no 1, p. 18.

[7] Frédéric FAVRE, « Web 3.0 : Interrogation intelligente », 2007, p. 149

[8] JPL
http://www.swi-prolog.org/packages/jpl/java_api/ consulté le 10 avril 2017.

[9] WIKIPEDIA. Web sémantique Web de données
http://fr.wikipedia.org/wiki/Web_s%C3%A9mantique consulté le 3 mai 2017

https://fr.wikipedia.org/wiki/Web_des_donn%C3%A9es consulté 3 mai 2017

[10] TUTORIAL PROTEGE-OWL. *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools*
http://www.coode.org/resources/tutorials/ProtegeOWLTutorial.pdf consulté 12 mai 2017

[11] W3C. SPARQL Query Language for RDF
http://www.w3.org/TR/rdf-sparql-query/ consulté 15 mai 2017

[12] W3C. Web Ontology Language (OWL)
http://www.w3.org/2004/OWL/ consulté 15 mai 2017

[13] JENA. Jena – A Semantic Web Framework for Java
http://jena.sourceforge.net/ consulté 16 mai 2017

[14] Langage de requêtes.
https://fr.wikipedia.org/wiki/Langage_de_requ%C3%AAte consulté 16 mai 2017

[15] Semantic Web Stack, définitions des couches
https://www.supinfo.com/articles/single/3358-web-semantique consulté 16 mai 2017

.

ANNEXES

– Règles de calcul de complexité

• Simplification

On calcule le temps d'exécution en effectuant les simplifications suivantes :

- on oublie les constantes multiplicatives (elles valent 1) ;
- on annule les constantes additives ;
- on ne retient que les termes dominants ;

Exemple : Soit un algorithme effectuant $g(n) = 4n^3 + 5n^2 + 2n + 3$ opérations :

- on remplace les constantes multiplicatives par 1 : $1n^3 + 1n^2 + 1n + 3$;
- on annule les constantes additives : $n^3 + n^2 + n + 0$;
- on garde le terme de plus haut degré : $n^3 + 0$;
- et on a donc $g(n) = O(n^3)$.

• Combinaison de complexités

- les instructions de base prennent un temps constant, noté $O(1)$;
- on additionne les complexités d'opérations en séquence : $O(f1(n)) + O(f2(n)) = O(f1(n) + f2(n))$;
- même chose pour les branchements conditionnels ;
- dans les boucles, on multiplie la complexité du corps de la boucle par le nombre d'itérations ;

En résumé pour calculer la complexité d'un algorithme, il faut respecter les principes suivants :

- on calcule la complexité de chaque partie de l'algorithme ;
- on combine ces complexités conformément aux règles vues ci-dessus ;
- on simplifie le résultat grâce aux règles de simplifications vues ci-dessus : élimination des constantes et conservation du terme dominant.

- **JPL**

L'utilisation de JPL pour la connexion d'un programme Java à Prolog, requiert tout d'abord l'installation de SWI-Prolog. Ensuite, il faut accéder au dossier d'installation de SWI-Prolog, dans notre cas « C:\Program Files (x86)\swipl » et aussi au fichier jpl.jar qui se trouve dans « C:\Program Files (x86)\swipl\lib » et enfin déclarer les chemins dans le path.

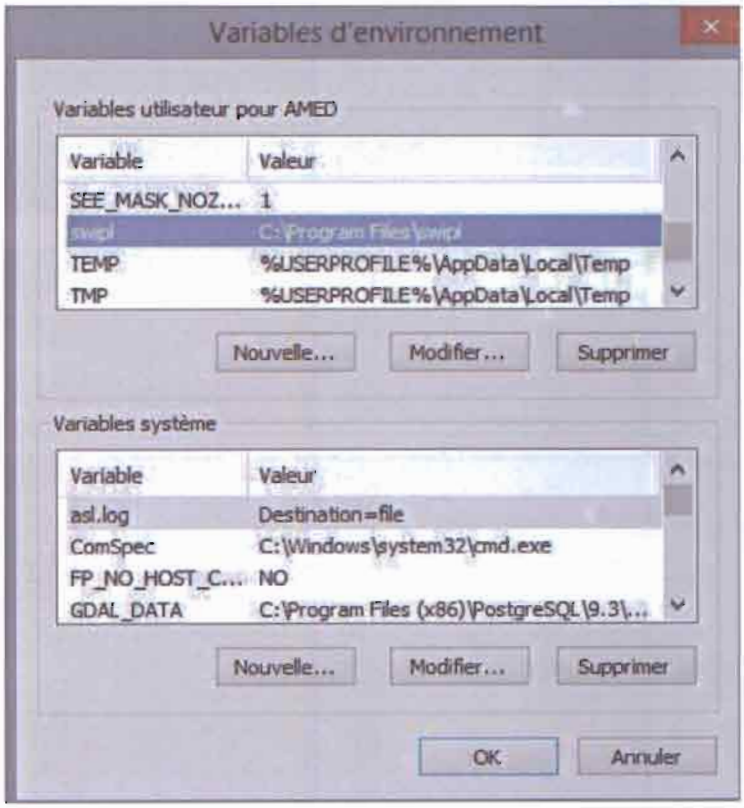


Figure 27. Déclaration dans le path des chemins pour JPL.

Une fois cette déclaration dans le path effectuée, on accède à Netbeans. Dans l'onglet « Tools→Librairies→New library », on crée une librairie nommée jpl par exemple. Puis dans cette nouvelle librairie, on charge le fichier jpl.jar à partir du bouton « Add Jar/Folder ».

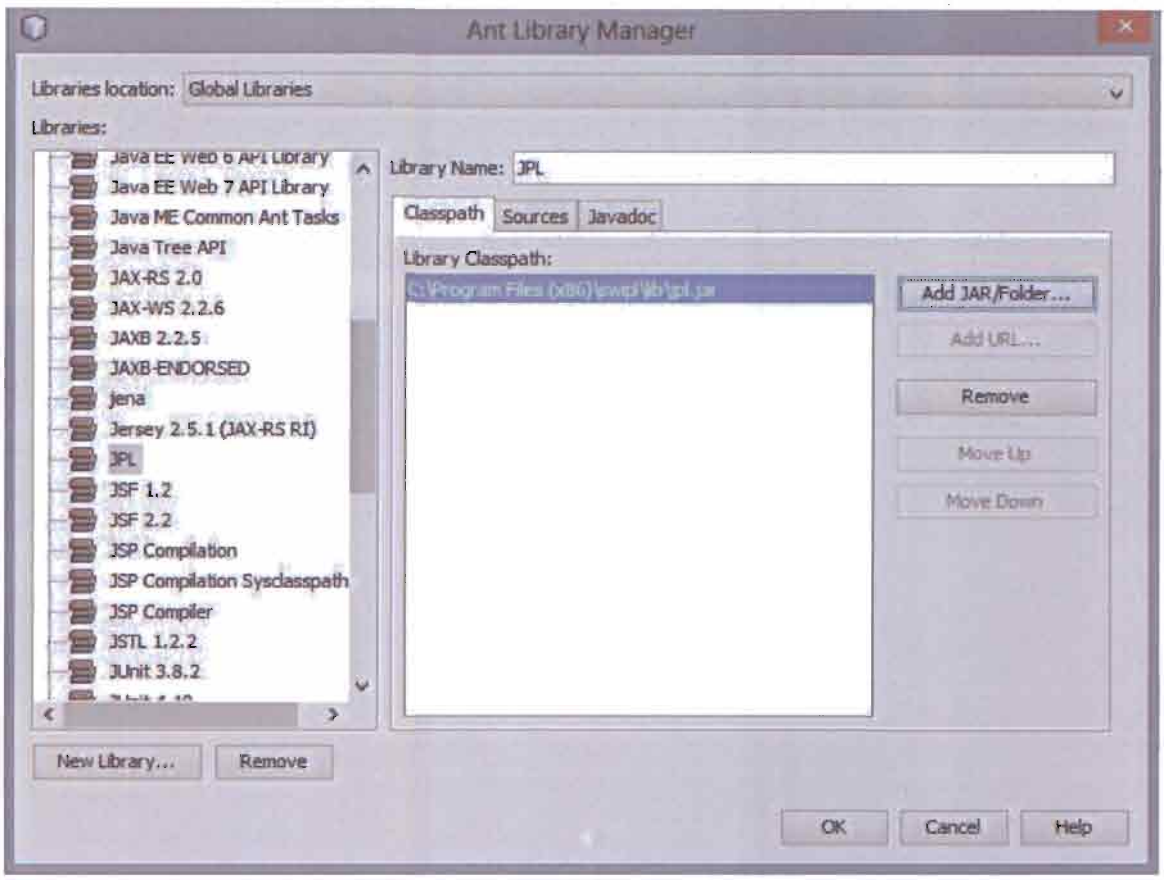


Figure 28. Création de la librairie « JPL »

Une fois la librairie créée, à partir du programme JAVA, dans l'onglet « librairies » à partir d'un clic droit puis sur le bouton « Add library », on choisit dans la liste des librairies, la librairie jpl.

– **JENA**

Jena est une librairie Java. Pour l'utiliser dans un projet, il est nécessaire de rajouter la librairie jena.jar dans le fichier « .classpath » du projet. Les logiciels de développement permettent d'effectuer l'ajout automatiquement. Sous NetBeans, il suffit d'aller dans le menu « Tools» puis dans « Libraries », cliquer sur le bouton « New Library ». Il faut nommer la nouvelle librairie « Jena » et sauvegarder. Ensuite dans l'onglet « Classpath », cliquer sur le bouton « Add JAR/Folder », naviguer vers les fichiers sources de Jena préalablement téléchargés (dossier /scr-lib) et ajouter tous les fichiers.

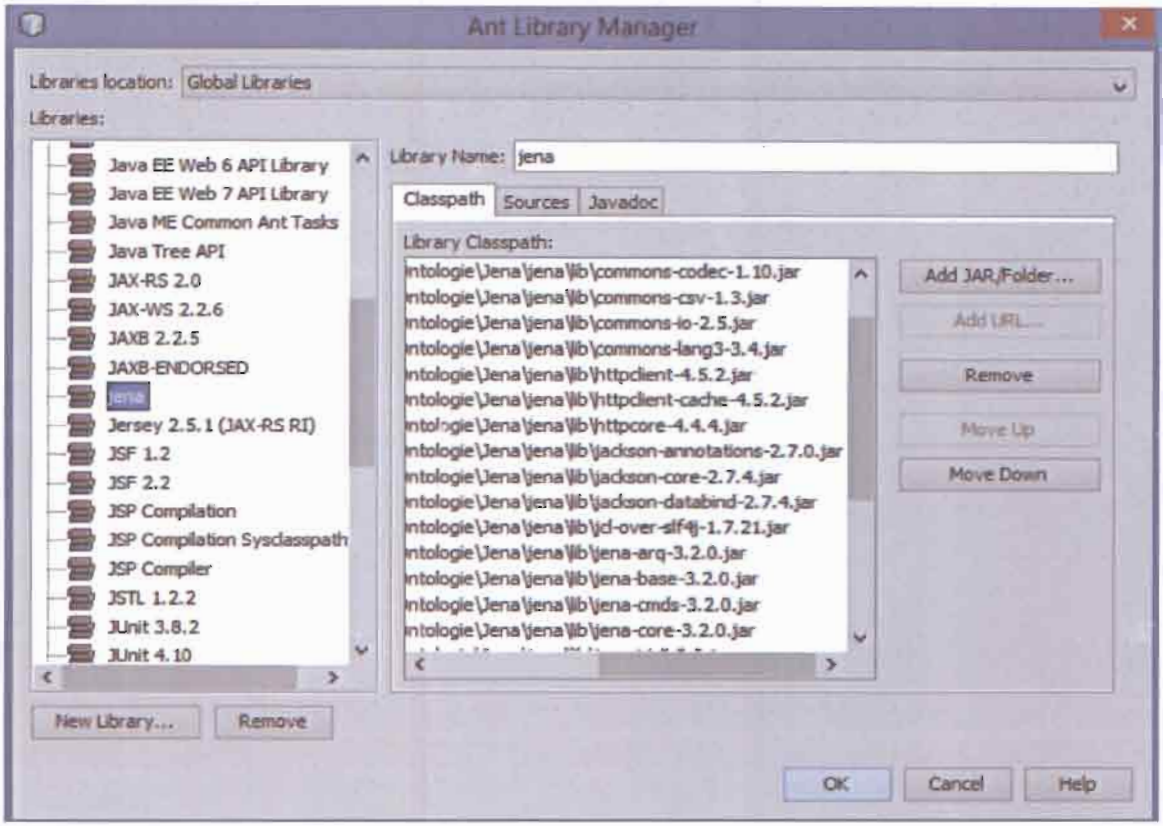


Figure 29 : Création de la librairie « Jena »

Après avoir enregistré la nouvelle librairie, il faut ouvrir le projet dans lequel on souhaite utiliser Jena, dans l'onglet « Librairies », à partir d'un clic droit, sélectionner « Add Library » et dans la liste choisir la librairie créée. Cette opération attribue la librairie Jena au projet.

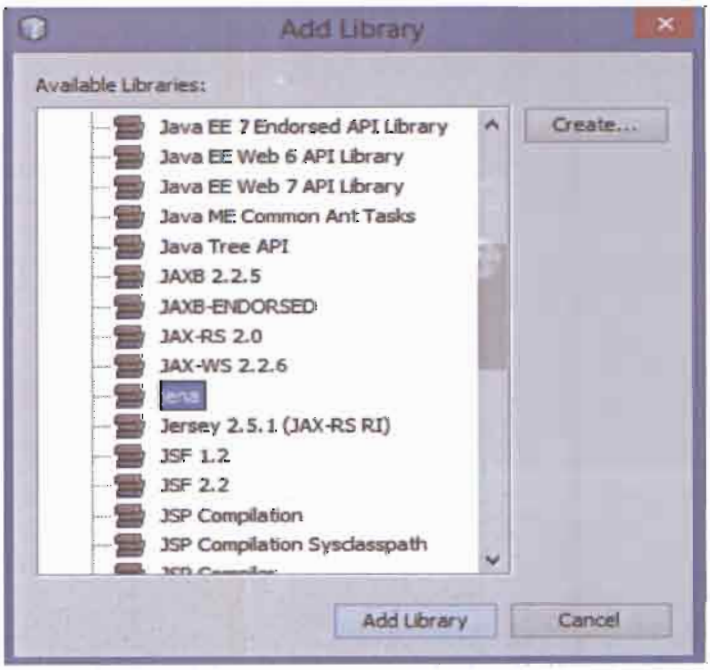


Figure 30. Ajout de la librairie « Jena » dans un projet JAVA