الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Master Académique (AC)

Thème :

Une démarche de spécification des besoins Et de conception pour la génération automatique d'un code java

Encadré Par :

Dr. Mohammed CHAOUI

Présenté par :

Nora Douakha

Bouchra Mehadjebia

Juin 2017

***A** notre encadreur.*

***A** nos Parents.*

***A** notre Famille.*

***A** nos Amis.*

Remerciements

En préambule à ce mémoire nous remercions ALLAH le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.
En second lieu, nous tenons à remercier notre encadreur ***Dr .Chaoui Mohammed****, son précieux conseil et son aide durant toute la période du travail.*

Ces remerciements vont tout d'abord au corps professoral et administratif de l'université 8 Mai 1945 de Guelma, plus particulièrement au corps professoral du département de l'informatique, pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

On n'oublie pas nos parents pour leur contribution, leur soutien et leur patience.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragées au cours de la réalisation de ce mémoire.

Résumé

Dans le processus de développement d'un programme ou bien un logiciel, les cahiers de charge ayant souvent des difficultés de compréhension pour que les développeurs puissent commencer le codage de lignes.

Il y a plusieurs démarches en génie logiciel tel que UML, qui facilitent le suivie de développement, mais pour arriver à développer correctement, il faut d'abord réaliser tous les diagrammes nécessaires. Dans ce sens, il y a deux points indispensables : une bonne conception avec des diagrammes corrects et un bon codage par les développeurs.

Notre proposition est sur cet axe où nous voulons proposer une démarche de spécification des besoins et de conception afin d'avoir un code Java directement sans passer à l'étape de développement.
Comme méthodologie de travail, nous partageons ce sujet en trois étapes principales : Étude de l'existant suivie de critique, Proposer une nouvelle démarche et faire leur conception et en fin la réalisation et l'implémentation de notre projet.

Mots clés : cahiers de charge, codage, génie logiciel, diagrammes, UML, code source, Java.

Abstract

In the process of developing a program or software, the Scope statement often having difficulty understanding so developers can begin row encoding.

There are several approaches in software engineering, Such as UML, Which facilitate the follow-up of development, but in order to develop correctly, it is necessary first to make all necessary diagrams. In This direction, there are two essential points: Good conception with correct diagrams and good coding by developers.

Our proposal is on this axis where we want to propose a specification and conception approach in order to have Java code directly without going to the development stage.
As a working methodology, we share this subject in three main stages: Study of the existent followed by criticism, suggest a new approach and make their conception. Finally the realization and implementation of our project.

Keywords: the Scope statement, encoding, Software engineering, diagrams, UML, source code, Java.

الملخص

كثيرا ما يواجه صعوبات في الفهم لكي يتمكن المطورين بدء دفتر الاحتياجات في عملية تطوير البرنامج أو برام كتابة أسطر البرنامج.

هناك العديد من النهج في هندسة البرمجيات مثل لغة النمذجة الموحدة والتي تسهل عملية متابعة تطوير البرنامج، ولكن للوصول الى تطوير بشكل صحيح فإنه يجب أولا تحقيق المخططات اللازمة. من هذا المنطلق نستنتج أن هناك نقطتين أساسيتين : التصميم الجيد مع المخططات الصحيحة والبرمجة الجيدة من قبل المطورين.

اقتراحنا هو على هذا المحور حيث نريد اقتراح نموذج جديد من أجل تحديد متطلبات عمل وكذا تصميمه من أجل الحصول على برنامج جافا مباشرة دون اجتياز كل مرحلة من مراحل تطوير برنامج هذا العمل.

منهجية العمل هي كالتالي : الموضوع ملخص في ثلاث مراحل رئيسية: دراسة الأعمال الحالية مع النقد، اقتراح نموذج جديد وكذا تصميمه وفي النهاية تطوير المقترح من أجل تحقيق هدف المشروع.

كلمات مفتاحية : بيان الاحتياجات، البرمجة، هندسة البرمجيات، الرسوم البيانية، لغة النمذجة الموحدة، كود المصدر، جافا.

Table des matières

Table des figures

Liste des Tableaux

LISTE DES ABREVIATIONS

Workflow : processus d'automatisation des tâches

COBOL: Common Business Oriented Language

BPM : Business Process Management

BPMN : Process Model and Notation

AWP : Active WEBDEV Page

Ajax: acronym d'*Asynchronous Javascript and Xml*

SDLC : The systems development life cycle

Introduction générale

Contexte et cadre de la recherche

Parfois, être un programmeur est difficile. Peut-être que vous faites la même chose plusieurs fois. Si vous voulez accéder à une base de données dans un programme, vous devez écrire un code semblable au code dans votre dernier programme que vous avez écrit pour accéder à une base de données.

Les interfaces utilisateur sont un autre exemple. Il est souvent douloureux d'écrire du code en Java pour créer l'interface utilisateur exacte que vous voulez. Généralement, après avoir écrit le code, puis le compiler, vous remarquerez que l'interface utilisateur est rien comme prévu, et aucun de vos boutons êtes où vous vous attendez qu'ils soient. Bien sûr, quand vous écrivez l'interface utilisateur (interfaces utilisateur) vous faites généralement la même chose encore et encore.

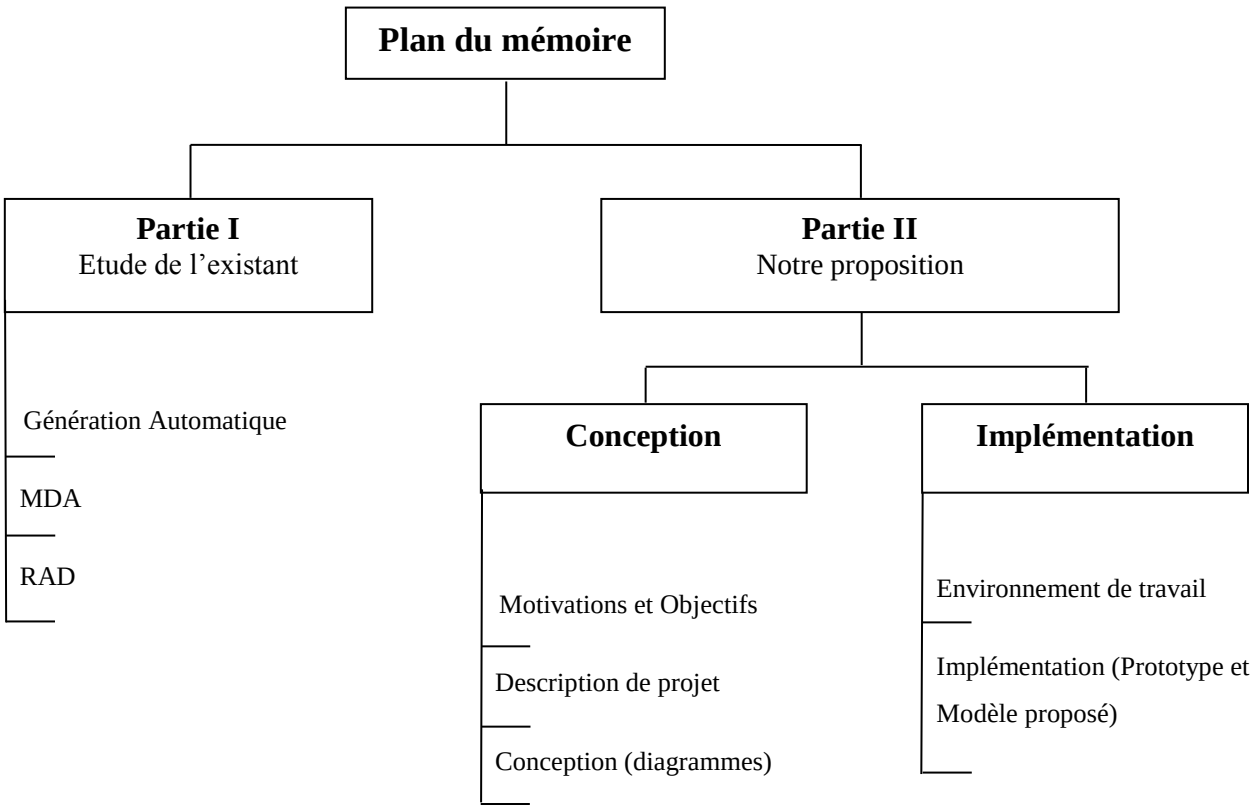
En fait, la plupart du temps, il est facile de se perdre dans la vue syntaxique ou plate-forme de détails spécifiques et lâche de la grande image. Ceci est où la génération automatique de code entre en jeu.

La Génération automatique de code fait référence à l'aide de programmes pour générer du code que le développeur aurait autrement à écrire. Comme un bonus supplémentaire, en utilisant la génération automatique de code crée des logiciels compatibles, une meilleure qualité de façon plus productive et à un niveau d'abstraction plus élevé, puis le codage manuel des projets.

Objectif et approche

Élaborer une démarche de spécification des besoins Et de conception pour la génération automatique d'un code Java afin d’avoir un code Java directement sans passer à l’étape de Développement (En offrant à n’importe quel utilisateur une nouvelle approche de programmation simple et facile a utilisé). Puis, accélérer le processus de développement des logiciels.

Plan du mémoire



Le présent mémoire est organisé en deux grandes parties. La première est consacrée à l'état de l'existant où est passée en revue la littérature relative aux technologies traitées. En effet, un état de l'existant recensant La génération automatique de code est présenté dans le premier chapitre. Dans le deuxième chapitre, nous décrivons L'approche MDA. Dans le chapitre troisième, nous allons étudier La technologie RAD.

Dans la deuxième partie, nous allons proposer notre proposition, donc effectivement le quatrième chapitre sera consacré à la conception de notre approche et le cinquième à la réalisation et l'implémentation de cette approche.

PARTIE – I : ETUDE DE L’EXISTANT

Chapitre 1 La génération automatique de code

1. Introduction

La génération d'une application à partir de son modèle par génération de code est un processus en deux étapes. La première étape est une transformation du modèle en langage de programmation de haut niveau(C, C++, java, etc.).

La seconde étape consiste en la compilation de ce programme en binaires exécutable qui font l'application par utilisation d'un compilateur support du langage ciblé lors de la première étape.

Les types de générateurs sont multiples, selon les besoins, mais la fonctionnalité principale reste la même : produire du code automatique, et soulager le développeur de certaines lourdeurs de mise en place. [1]

Dans ce chapitre, nous allons introduire La génération automatique de code, et quelques exemples sur les générateurs existants, ainsi, quelques observations sur ces générateurs.

2. La génération automatique de code

La génération automatique de code fait référence à l'utilisation de programmes pour générer du code que l'utilisateur aurait sinon à écrire eux-mêmes. [2]

2.1 Qu'est-ce qu'un générateur de code ?

Un générateur de code est un outil ou une ressource qui génère un type particulier de code ou un langage de programmation informatique. Cela a de nombreuses significations spécifiques dans le monde de l'informatique, dont beaucoup sont liés aux processus parfois complexes de conversion de la syntaxe de programmation humaine au langage machine qui peut être lu par un système informatique.[3]

2.2 Les avantages de La génération automatique de code par rapport au codage traditionnel

- Haute qualité
- Cohérent
- Productif
- Codage abstraite [2]

2.3 Les Inconvénients de La génération automatique de code

- Taille du code (Pas un gros problème)
- Intégration avec le code hérité
- Cohérence entre le modèle et le code (tentation de modifier le code plutôt que de réviser le modèle et de le générer de nouveau). [4]

2.4 Classification

- **Code Munger** : Génère un nouveau code à partir du code existant.
- **Expandeur de code en ligne** : Développe un code spécial dans un programme.
- **Générateur de code mixte** : Insère un nouveau code dans le code existant.
- **Générateur de classes partielles** : Crée un code cadre à partir d'un modèle. Dire des déclarations de fonction vides.
- **Générateur de niveau :** Crée un code complet à partir d'un modèle.
- **Langue du domaine :** Une nouvelle langue pour décrire un type spécifique de problème.[2]

3. Générateurs existants

3.1 Générateur de code avant conception (Gap)

Définition :

Le générateur automatique de programmes (GAP) est un langage de programmation destiné à la gestion. [7]

Principe : produire automatiquement les programmes à partir d'une description abstraite des données.

A partir d'un modèle mathématique (ensembliste) décrivant les données, leurs relations et les principales fonctions de calcul, le système GAP produit automatiquement le logiciel (programmes serveurs et écrans). Les fonctions de validation des dossiers (workflow), d'identification des utilisateurs et de restriction de leurs droits d'accès aux données sont intégrées dans le système.

Les calculs complexes et les écrans qui leurs correspondent doivent être programmés manuellement.

Cette approche qui consiste à générer les programmes à partir d'une spécification formelle du résultat attendu a déjà été utilisée avec succès dans le domaine des automates industriels.
Elle n'avait que peu été explorée pour les applications de gestion. Le système GAP comble cette lacune. [8]

Utilisations :

Le système peut être utilisé pour aider l'analyse en produisant itérativement des prototypes destinés à affiner l'expression des besoins.

Les équipes de développement peuvent augmenter leur productivité en produisant automatiquement l'essentiel des programmes d'un projet. Elles peuvent aussi réutiliser les modèles de données de différents projets. La réutilisation des composants logiciels, souvent souhaitée mais peu pratiquée, trouve avec le système GAP une application très concrète.

Ce système facilite la reprise des applications existantes. Des programmes spécifiques peuvent être écrits pour produire le modèle des données à partir des structures de données d'une application (COBOL par exemple). Les traitements complexes doivent, par contre, être repris manuellement.

Les directions informatiques qui souhaitent (ré) homogénéiser leur système d'information peuvent utiliser ce système pour leurs nouveaux développements. Les programmes générés sont auto-documentés par le modèle des données et leurs structures sont identiques, ce qui facilite leur maintenance et la polyvalence des équipes techniques. Par ailleurs, ils sont conçus pour permettre l'urbanisation du système d'information en l'organisant en composants applicatifs homogènes dialoguant par une interface standardisée. [8]

3.2 Générateur de code à partir des besoins avec conception (*Bizagi)*.

Définition :
Bizagi BPM Suite gère le cycle de vie complet d'un processus métier : Modèle, Construction Et Exécuter. Chacune de ces étapes est gérée par différents produits de notre Suite Qui permettent, en utilisant un environnement graphique et dynamique, la construction d'un Solution basée sur les processus.

Figure 1.1 : *Présentation de principe de bizagi*

1) Principe :

1. Cartes de processus de conception :

La première étape pour créer des solutions Bizagi est de concevoir un flux de processus à l'aide de Bizagi Modélisateur Ce produit de Bizagi BPM Suite est un modèle de Outil de documentation. Ce produit vous permet de concevoir, documenter et Simuler les processus d'affaires, de manière simple et rapide, en utilisant le BPMN, un format accepté dans le monde entier pour la modélisation des processus.

2. Construire des applications de processus :

Une fois la phase de conception du processus terminée, l'étape suivante consiste à automatiser Vos processus.

Bizagi Studio est le produit de notre suite BPM qui fournit Environnement de construction pour transformer vos cartes de processus en applications en cours d'exécution Sans code.

Bizagi Studio est un outil gratuit qui fournit un environnement collaboratif multi-utilisateurs, Conçu pour construire et conserver toutes les informations nécessaires à l'exécution du processus : flux Diagramme, données de process, interface utilisateur, règles métier, etc.

Bizagi Studio propose un ensemble de fonctionnalités pour générer graphiquement un modèle associé à Un processus commercial ; Un assistant convivial vous guide à travers toutes les étapes nécessaires pour Transformer vos cartes de processus conçues dans Bizagi Modeler, en applications en cours d'exécution(Workflows).

3. La dernière étape est l'exécution de vos applications

Le modèle résultant de la phase de construction dans Bizagi Studio est stocké dans le serveur Et est interprétée et exécutée en production par le troisième produit de la Bizagi BPM Suite: Moteur Bizagi. Ce produit est basé sur une collection de Composants qui offrent toutes les

fonctionnalités nécessaires à une entreprise efficace Gestion des processus dans l'organisation (portail de travail, BAM, règles d'affaires,Moteur d'intégration, etc.).

Bizagi Engine gère l'exécution optimale des différentes tâches et activités Qui composent le processus d'affaires. Il contrôle et vérifie que toutes les tâches sont Par la personne ou la ressource appropriée et selon Les politiques commerciales de l'entreprise, ses objectifs et d'autres règles fondamentales. [5]

3.3 Générateur de code après conception (Acceleo)

Définition : est un outil de génération de code sous Eclipse. Il permet de concevoir des modules de génération de code dans un langage choisi par le développeur, à partir d'un ou plusieurs modèles, et fournit aussi des modules de génération de code prêts à être utilisés (UML vers Java, UML vers C#, etc.).[6]

Principe :

Acceleo fournit un moteur qui exécute des modules (fournis par Acceleo ou fournis par des tierces parties) sur les modèles (UML, Merise, ..) .(figure2) pour générer le code dans le langage supporté par le module. Par exemple, un plugin Eclipse open source « UML to Java Generator » permet la génération de code java depuis un modèle UML.(figure3)

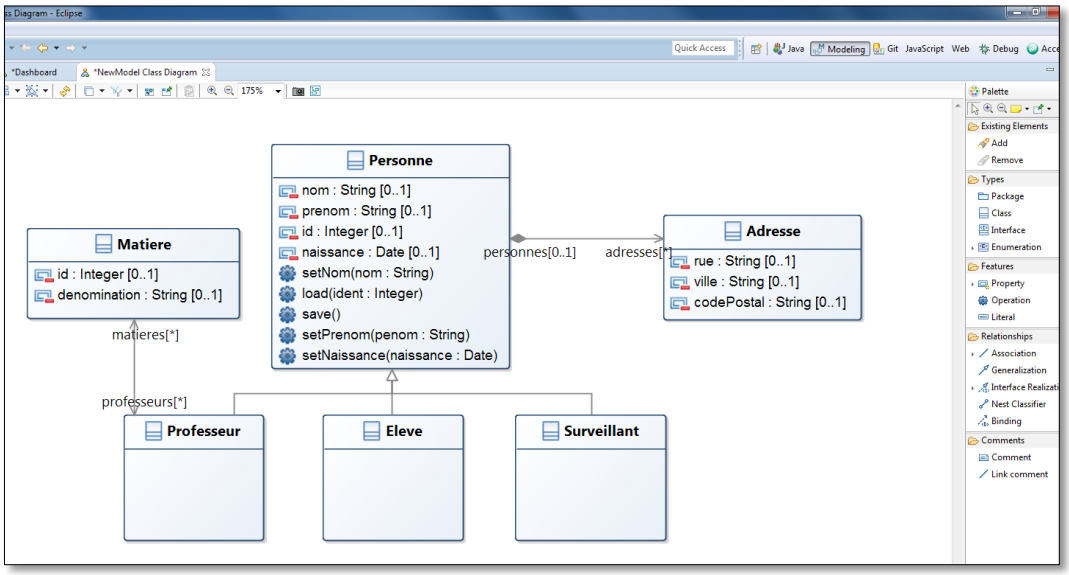


Figure 1.2 : *Représentation d'un Model UML*

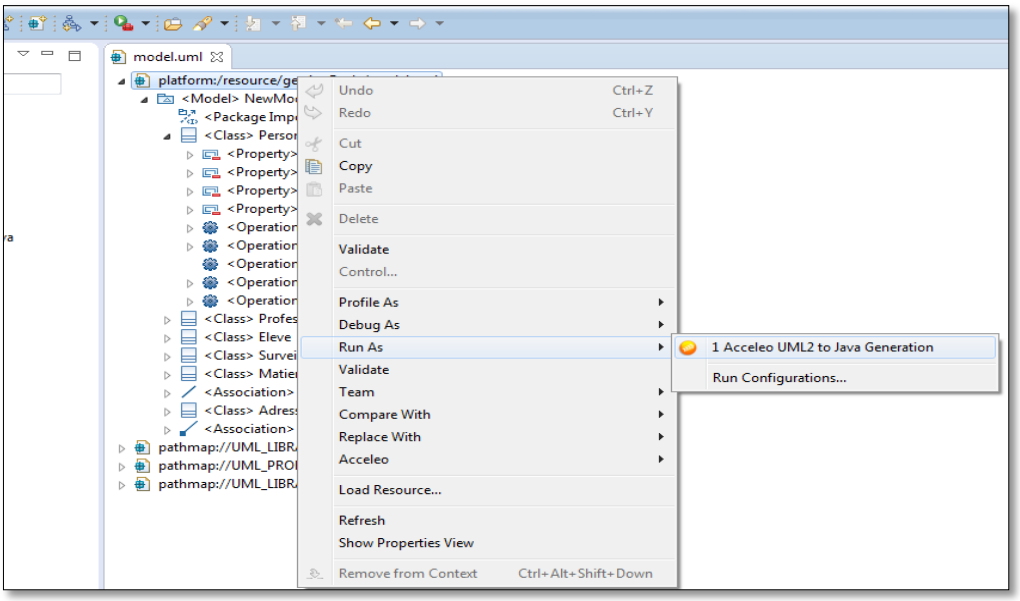


Figure 1.3 *: La Génération de Code java depuis un modèle UML*

À l'issue de cette génération automatique, chaque entité du diagramme de classes est devenue une Classe, Interface ou Énumération selon le type spécifié lors de la modélisation UML.(figure 4)

Le package renfermant ces classes se trouve dans un projet nommé :**Org.obeonetwork.uml.to.java** . **[9]**

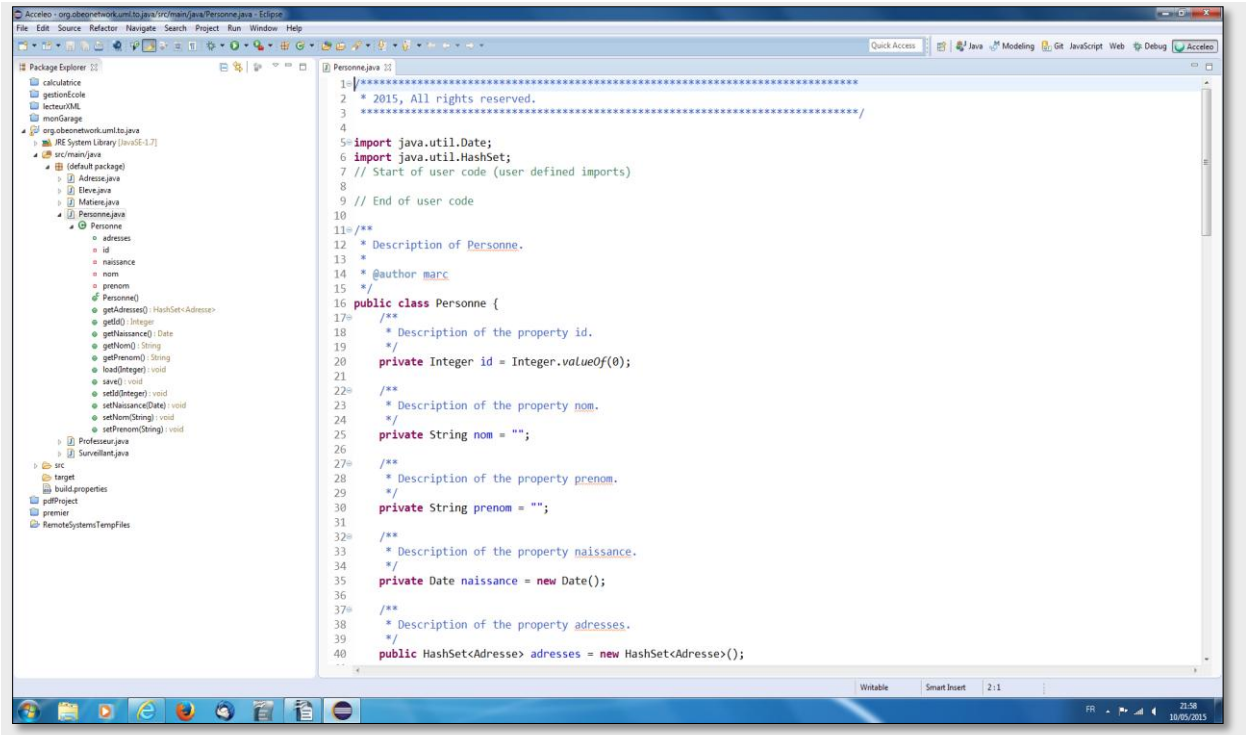


Figure 1.4 *: Le résultat de La Génération de Code java depuis un modèle UML*

4. Observations sur les générateurs existants

Générateur	*Avantage*	*Inconvénient*
Système GAP	Des logiciels moins coûteux et plus fiables	Un système uniquement destiné à produire rapidement des logiciels de gestion fiables
Bizagi	Application professionnelle puissante avec une très bonne interface facile à utiliser avec une bonne documentation	Un produit qui semble intéressant mais nous ne l'avons pas assez testé pour savoir s'il est facile d'exécuter et implémenter un processus
Acceleo	• interopérabilité des méta-modèles d'entrée (UML 1 / UML 2 / DSL conformes à EMF). • syntaxe arborescente dédiée à la manipulation de modèles. • personnalisation de la génération par templates. • indépendance du langage généré.	• Limites de stockage Accelo limite actuellement le montant du stockage de base de données tel qu'indiqué dans le bon de commande concerné. • Limitation de responsabilités : Accelo ne sera pas responsable de la de profits ou de dommages spéciaux, indirects, accessoires ou consécutifs indépendant la forme d'action, même si Accelo est a visé de la possibilité de tels dommages.

***Tableau 1.1** : Tableau de Comparaison des outils de génération de code*

5. Conclusion

En conclusion, la génération automatique de code peut améliorer la qualité et la cohérence d'un programme de manière productive. La génération automatique de code est également suffisamment abstraite pour être utilisée à des fins multiplateformes. Compte tenu de la grande variété d'outils de génération automatique de code, le seul problème est de trouver le bon programme pour le travail. Bien sûr, nous pourrions aussi construire leur propre générateur de code. Nous décrivons, dans le chapitre qui suit, le principe de l'approche MDA.

Chapitre 2 L'approche MDA

1. Introduction

Les technologies sont en constante évolution. Afin de bénéficier des avancées technologiques, il est nécessaire d'adapter les applications à ces technologies. Or cette opération coûte cher aux entreprises, car il est souvent nécessaire de réécrire le code entièrement. Lorsqu'il n'existe pas de capitalisation des fonctions de l'application et que le développement repose généralement sur le code source, la séparation des préoccupations apparaît comme la solution nécessaire au problème. Ainsi, spécifications fonctionnelles et spécifications techniques sont prises en compte séparément par l'approche MDA. [11]

Dans ce chapitre nous allons détailler le principe de l'approche MDA qui permet d'obtenir le code source de l'application par la génération automatique à partir des modèles de l'application.

2. l'architecture dirigée par les modèles – MDA

Après la technologie procédurale, la technologie objet et la technologie des composants, l'approche MDA (Model Driven Architecture) est un processus de l'ingénierie dirigée par les modèles (ou MDE pour Model Driven Engineering). Proposée par l'OMG (Object Management Group) en 2000, l'approche MDA est basée sur la séparation des préoccupations. Elle permet de prendre en compte, séparément, l'aspect métier et l'aspect technique d'une application, grâce à la modélisation. Le code source de l'application est obtenu par la génération automatique à partir des modèles de l'application. Les modèles ne sont plus seulement un élément visuel ou de communication, mais sont, dans l'approche MDA, un élément productif et le pivot du processus MDA. [10]

3. L'ingénierie dirigée par les modèles – IDM

Avant de parler plus en détail du sujet qui nous intéresse ici, à savoir l'approche MDA, énumérons les concepts sous-jacents sur lesquels est bâtie cette approche. La modélisation est la représentation de l'information à différents niveaux d'abstraction.

Dans le cas d'une application informatique, un modèle permet de représenter, à chaque étape du développement, certaines caractéristiques de l'application, car développeurs, analystes et architectes n'ont pas les mêmes besoins de connaissance de l'application et de son environnement. L'IDM est la mise en œuvre de la modélisation dans le domaine du développement logiciel.

Sur la base de la modélisation et des méthodologies de conception basées sur les modèles, l'OMG propose l'approche MDA, dérivée de l'IDM. De par ce fait, l'approche MDA est construite sur les mêmes bases que l'IDM, à savoir : le méta-modèle, le modèle et la transformation de modèle. Chacun de ces concepts est détaillé dans les sections qui suivent. [10]

4. Les Modèles

L'idée centrale de MDA est d'élaborer des modèles, d'abord d'analyse puis de conception, jusqu'au code, par transformations, dérivations et enrichissements successifs

L'OMG propose le langage déclaratif (à base de règles) "QVT" (Query/View/Transformation) pour exprimer les transformations de ces modèles. [10]

Les principaux modèles sont :

Modèle CIM- Computational Independant Model

Les modèles d'exigence CIM décrivent les besoins fonctionnels de l'application, aussi bien les services qu'elle offre que les entités avec lesquelles elle interagit. Leur rôle est de décrire l'application indépendamment des détails liés à son implémentation. Les CIM peuvent servir de référence pour s'assurer que l'application finie correspond aux demandes des clients.

Modèle PIM- Platform Independant Model

Les modèles PIM sont les modèles d'analyse et de conception de l'application. La phase de conception à cette étape du processus suppose l'application de Design pattern, le découpage de l'application en modules et sous-modules, etc. Le rôle des PIM est de donner une vision structurelle et dynamique de l'application, toujours indépendamment de la conception technique de l'application.

Modèle PM- Platform Model

Rarement utilisé, un PM décrit la structure, et les fonctions techniques relatives à une plateforme d'exécution (systèmes de fichiers, de mémoire, de BDD…) et précise comment les utiliser. Le PM est associé au PIM pour obtenir le PSM.

Modèle PSM- Platform Specific Model

Le PSM est le modèle qui se rapproche le plus du code final de l'application. Un PSM est un modèle de code qui décrit l'implémentation d'une application sur une plateforme particulière, il est donc lié à une plateforme d'exécution.

Code source

Représente le résultat final du processus MDA, le code source est obtenu par génération automatique (partielle ou totale) du code de l'application à partir du PSM. Le code source obtenu peut toujours être enrichi ou modifié manuellement. [10]

5. La transformation des modèles MDA

L'approche MDA précise quatre types de transformations pendant le cycle de développement, les modèles devenant de plus en plus concrets jusqu`à l`obtention du code Par transformations successives, le PIM, modèle de niveau le plus abstrait, est transformé en un PSM exécutable (ou code exécutable) Si la démarche MDA a été respectée, il est possible de générer un PSM, puis un PIM, à partir du code exécutable (rétro-ingénierie).[11]

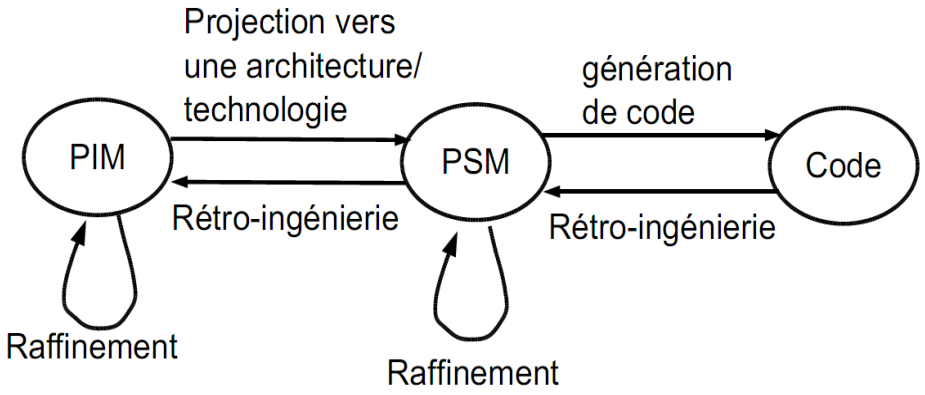


Figure 2.1 : *La transformation des modèles MDA*

6. La Différence Entre MDA & MDE

MDA	MDE ou IDM
➨ Placer les modèles au cœur des processus ➨ Séparation des spécifications fonctionnelles des spécifications techniques ➨ Modèles métiers : PIM ➨ Modèles spécifiques : PSM ➨ Basée sur les langages standards : UML, MOF, QVT... ➨ Vers l'ingénierie dirigée par les modèles	➨ Généralise l'idée du MDA ➨ Propose de structurer l'espace des modèles dans toutes ses dimensions ➨ Problème de la mise en relation des différents modèles ➨ Vise à faire des modèles des artefacts manipulables

Tableau 2.1 : *La Différence Entre MDA & MDE*

7. Les avantages de MDA

L`initiative d'architecture dirigée par les modèles de l'OMG "Model Driven Architecture" (MDA) est motivée par le besoin de réduire les tâches de conception des applications (nécessitées, en autre, par l`évolution constante des technologies informatiques).

Puisque les modèles sont plus pérennes que les codes, ils permettent de :

- conserver les exigences métiers (échanges entre analystes et donneurs d'ordre)
- réutiliser les choix d`architecture et de codage (échanges entre analystes et programmeurs)
- assurer l`intégrité et la cohérence entre les phases du projet (tests) [11]

8. Les inconvénients de MDA

Bien que la démarche comporte des avantages indéniables, elle peut en rebuter certains :

- En effet, pour développer avec la démarche MDA, cela nécessite des architectes expérimentés qui connaissent et maîtrisent le développement orienté objet ainsi que les patrons de conception. Et c'est une denrée rare au vu de ce sondage où seulement 7 % des sondés comprennent les enjeux de l'approche MDA contre 59 % qui n'ont jamais entendu parler de MDA.
- De même pour la nouvelle version d'UML, seulement 6% comprennent les enjeux de la version 2 d'UML contre 46 % qui ne connaissent pas UML 2.0 (SDTime2002). D'autant plus qu'il n'est pas établi, qu'UML soit un langage efficace pour la programmation.
- Le MDA reste compliqué à mettre en œuvre. Pour faciliter son utilisation, il est nécessaire d'avoir des outils d'automatisation. Or aujourd'hui, ces outils ne communiquent pas tous entre eux et certains sont spécifiques à une plate-forme en particulier.
- Il faut noter aussi que les plates-formes pourraient évoluer plus rapidement que les outils de transformation entraînant ainsi un manque d'interopérabilité.
- De plus, le passage automatisé de PIM vers PSM est loin d'être achevé, d'autant que la distinction entre les deux n'est pas franche. Même s'il existe des outils qui permettent de faire ce passage automatiquement, le passage du PIM vers le code n'est pas complet, avec seulement 80 % de code généré à l'heure actuelle. [13]

9. Exemple :

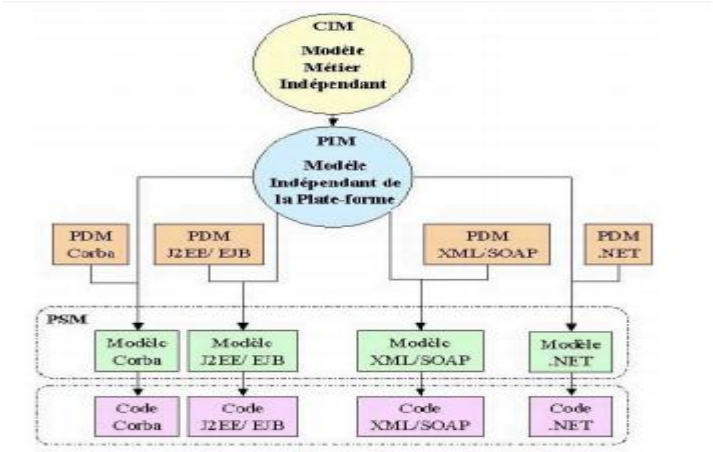


Figure 2.2 : *Utilisation des modèles pour réaliser une application. [11]*

10. Conclusion

Nous avons abordé dans ce chapitre l'architecture dirigée par les modèles (*Model Driven Architecture*) de l'OMG. L'approche MDA se base sur les mêmes principes que l'ingénierie dirigée par les modèles, et comme nous l'avons brièvement vu, ce n'est pas la seule approche de génie logiciel à adopter les modèles dans son processus de développement.

Le point commun entre toutes ces approches est le fait que les modèles représentent le pivot du processus de conception.[10] . Nous allons donc étudier, dans le chapitre suivant, la technologie RAD qui se base aussi sur les modèles de programmation pour générer automatiquement les applications.

Chapitre 3 La technologie RAD

1. Introduction

La génération automatique d'applications a évolué vers les phases hautes du processus de développement logiciel pour, dès en amont, transformer les besoins exprimés en langage naturel en artefacts logiciels.

Les usages les plus marqués sont le dessin d'interfaces utilisateur via les environnements de développements intégrés (IDE ou Integrated Development Environment en anglais) capables de produire le code construisant mais surtout contrôlant ces interfaces utilisateur dans les applications déployées. Cette approche est pour l'essentiel à l'origine de l'acronyme RAD (Rapid Application Development) .[15]

2. Présentation de technologie RAD

Le RAD (Rapid Application Development) est une technologie permettant de créer en quelques clics une application complète (ou un site complet). Utilisée de manière efficiente cette puissante fonctionnalité permet de gagner du temps lors de la construction de l'IHM.

Le RAD se base sur des modèles de programmation pré-établis (appelés Patterns), permettant de générer l'application voulue. Il est également possible de créer entièrement ses propres "patterns" pour générer automatiquement une application personnalisée : vous décidez du code généré ainsi que du positionnement des champs. « . Windev permet de créer entièrement ses propres "patterns".»

Le pattern RAD Imperso est un pattern RAD personnalisé qui utilise des modèles de fenêtres et des classes des fenêtres pour produire une application parfaitement intégrée au framework wiZtudio. Le pattern RAD Imperso confère également aux applications un nouveau look moderne et convivial. [16]

3. Définition de model RAD :

Le modèle RAD est un modèle de développement rapide d'applications. C'est un type de modèle incrémental. Dans le modèle RAD, les composants ou les fonctions sont développés en parallèle comme s'il s'agissait de mini projets.

Les développements sont en boîtes de temps, livrés et ensuite assemblés en prototype fonctionnel. Cela peut rapidement donner au client quelque chose à voir et à utiliser et à fournir des commentaires concernant la livraison et leurs exigences.

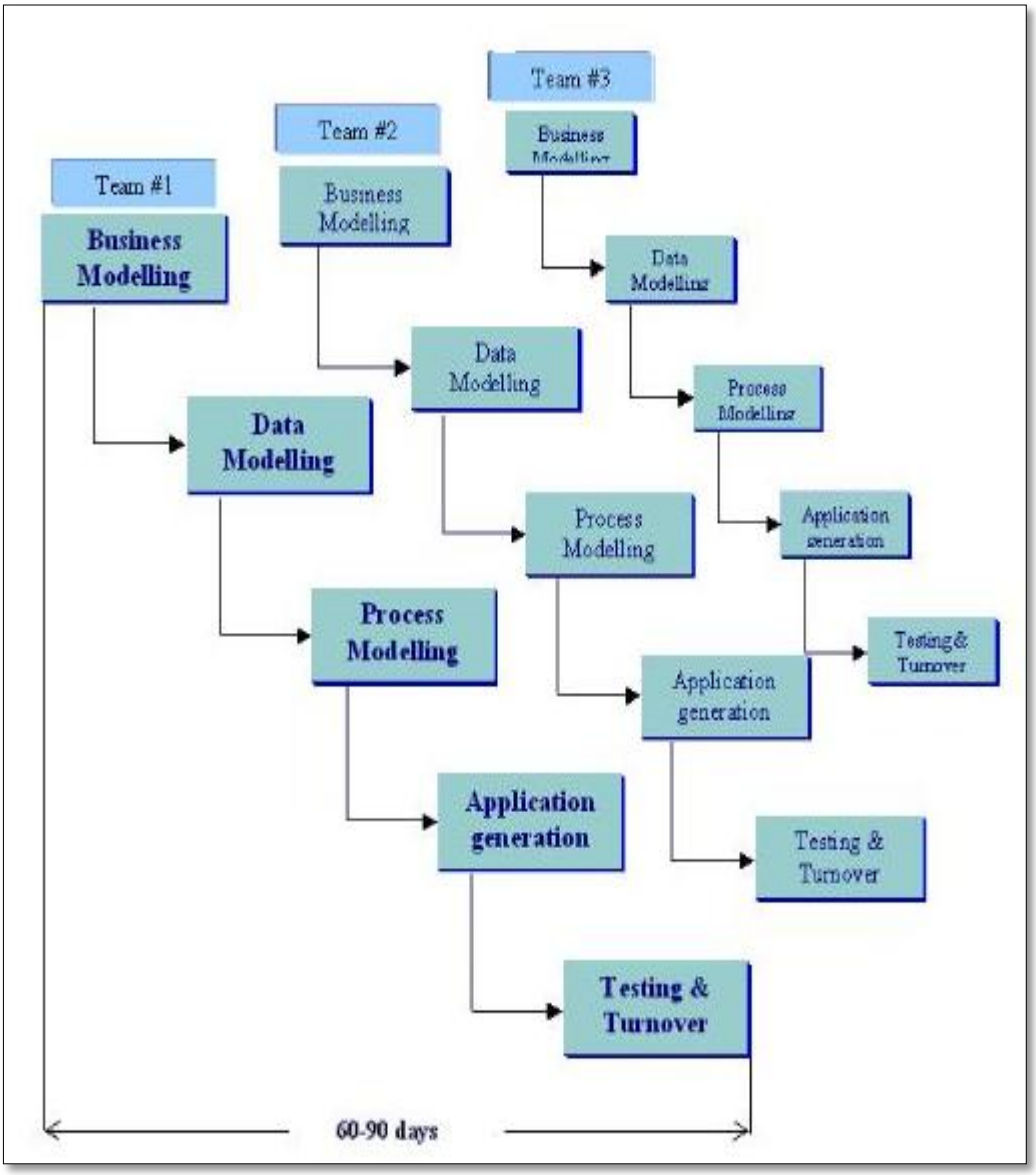


Figure 3.1 : *RAD modèle*

Le modèle RAD a les phases suivantes :

Business Modeling : Le flux d'information entre les fonctions de l'entreprise est définie par la réponse à des questions telles quelles informations le moteur du processus d'affaires, ce que l'information est générée, qui le génère, d'où vient l'information, l'identité de ce processus et ainsi de suite.

Modélisation des données : Les informations recueillies auprès de modélisation d'entreprise est transformée en un ensemble d'objets de données (entités) qui sont nécessaires pour

soutenir les activités. Les attributs (caractère de chaque entité) sont identifiés et la relation entre ces objets de données (entités) est définie.

Modélisation des processus : L'objet de données définies dans la phase de modélisation des données sont transformées pour atteindre le flux d'informations nécessaires pour mettre en œuvre une fonction de gestion. Descriptions de traitement sont créées pour ajouter, modifier, supprimer ou retirer un objet de données.

Applications de nouvelle génération : des outils automatisés sont utilisés pour faciliter la construction du logiciel, et même ils utilisent les techniques GL 4e.

Essais et Chiffre d'affaires : La plupart des composants de programmation ont déjà été testés depuis RAD accent réutilisation. Cela réduit le temps de test global. Mais de nouveaux composants doivent être testés et toutes les interfaces doivent être pleinement exercées. [15]

4. Les modèles RAD disponibles

4.1 Application WINDEV

Les patterns RAD proposés par défaut sont les suivants :

- **RAD Simple :**

Ce pattern permet la génération d'une application complète. Cette application est composée de fiches, tables et zones répétées. Le code utilisé est le plus simple possible. Il est possible d'appliquer n'importe quelle charte graphique.

- **Pattern Antakara :**

Ce pattern permet de créer une application proposant des fonctionnalités avancées comme une liste des tâches, un calendrier, … Cette application propose également un menu constitué d'images animées.

- **Pattern Pure :**

Ce pattern permet de créer une application utilisant une interface à utiliser de préférence sur les écrans larges.

- **Pattern Jet'Tames :**

Ce pattern permet de créer une application proposant en standard l'envoi d'emails. Le menu de l'application utilise un carrousel d'images.

- **Pattern RAD compatible 11 :**

Ce pattern permet de générer une application correspondant aux applications générées par la version 11. [17]

4.2 Application WINDEV Mobile

Les patterns RAD proposés par défaut sont les suivants :

- **RAD Simple :**

Ce pattern permet la génération d'une application complète. Cette application est composée de fiches, tables et zones répétées. Le code utilisé est le plus simple possible. Il est possible d'appliquer n'importe quelle charte graphique.

Ce pattern est proposé si le projet est réalisé pour un Mobile.

- **RAD Simple pour Smartphone :**

Ce pattern permet la génération d'une application complète. Cette application est composée de fiches, tables et zones répétées. Le code utilisé est le plus simple possible. Il est possible d'appliquer n'importe quelle charte graphique.

Ce pattern est proposé si le projet est réalisé pour un Smartphone.

- **Pattern RAD compatible 11 :**

Ce pattern permet de générer une application correspondant aux applications générées par la version 11.

4.3 Site WEBDEV

Les patterns RAD proposés par défaut sont les suivants :

- **Pattern RAD AWP :**

Ce pattern est destiné à la génération d'un site AWP. Il propose les fonctionnalités suivantes : tables Ajax, zones répétées, génération d'états au format PDF, gestion des menus, gestion du référencement, gestion de cellules déplaçables.

- **Pattern Intranet :**

Ce pattern est destiné à la génération d'un site Intranet/Extranet. Il propose les fonctionnalités suivantes : tables Ajax, génération d'états au format PDF, gestion des menus sous forme d'onglets.

- **Pattern RAD compatible 11 :**

Ce pattern permet de générer une application correspondant aux applications générées par la version 11.[15]

5. Quand utiliser le modèle RAD

RAD devrait être utilisé lorsqu'il est nécessaire de créer un système qui peut être modulé en 2 à 3 mois.

Il devrait être utilisé s'il existe une forte disponibilité de concepteurs pour la modélisation et que le budget est suffisamment élevé pour supporter leur coût ainsi que le coût des outils automatisés de génération de code.

Le modèle RAD SDLC devrait être choisi uniquement si des ressources avec des connaissances commerciales élevées sont disponibles et il est nécessaire de produire le système dans un court laps de temps (2-3 mois).

6. Avantages du modèle RAD :

- Réduction du temps de développement.
- Augmente la réutilisation des composants
- Des commentaires initiaux rapides se produisent
- Encourage les commentaires des clients
- L'intégration dès le début résout beaucoup de problèmes d'intégration.

7. Inconvénients du modèle RAD :

- Dépend de fortes performances en équipe et individuelles pour identifier les besoins de l'entreprise.
- Seul le système qui peut être modulaire peut être construit en utilisant RAD
- Nécessite des développeurs / concepteurs hautement qualifiés.
- Grande dépendance à l'égard des compétences de modélisation
- Inapplicable à des projets moins coûteux car le coût de la modélisation et la génération automatisée de code est très élevé.[16]

8. Conclusion

Dans ce chapitre, nous avons introduit la technologie RAD qui est basé aussi sur les modèles pour gérer automatiquement les applications, qui présente l'objectif fondamental de notre approche. Dans le chapitre qui suit, nous allons réaliser la conception de cette architecture de notre nouvelle approche.

PARTIE – II : NOTRE PROPOSITION

Chapitre 4 Conception

1. Introduction

Parmi les classifications de la génération automatique de code, notre travail est basé sur la génération de niveau (créer un code complet à partir d'un modèle).

La génération automatique de code à partir d'un modèle capturant graphiquement les spécifications du logiciel est de plus en plus couramment utilisée pour implémenter un logiciel, ou pour réaliser un prototypage rapide d'une application.

Il y a plusieurs démarches en génie logiciel tel que UML qui est conçue pour modéliser le système d'information. Elle est aussi accepté comme le langage standard de modélisation de l'analyse et la conception de systèmes logiciels. Mais pour arriver à développer correctement, il faut d'abord réaliser tous les diagrammes nécessaires.

Dans ce sens nous proposons une démarche de spécification des besoins et de conception afin d'avoir un code Java directement sans passer à l'étape de développement pour accélérer le processus de développement des logiciels.

2. Motivations et Objectifs

- Facilite le développement de logiciel à partir d'un modèle.
- Avoir un outil facile à utiliser par n'importe quel utilisateur (n'est pas besoin de développeur)
- Création automatique de code Java complet (Interfaces + code source de manipulation des données).
- En plus, Ce type de projet n'a jamais abordé au niveau de notre département.

3. Description de projet.

Notre démarche est composée de trois étapes (Figure 4.1) :

1) Identification des registres
2) Création automatique du code source des fenêtres (IHM)
3) Création automatique du code source de manipulation des données.

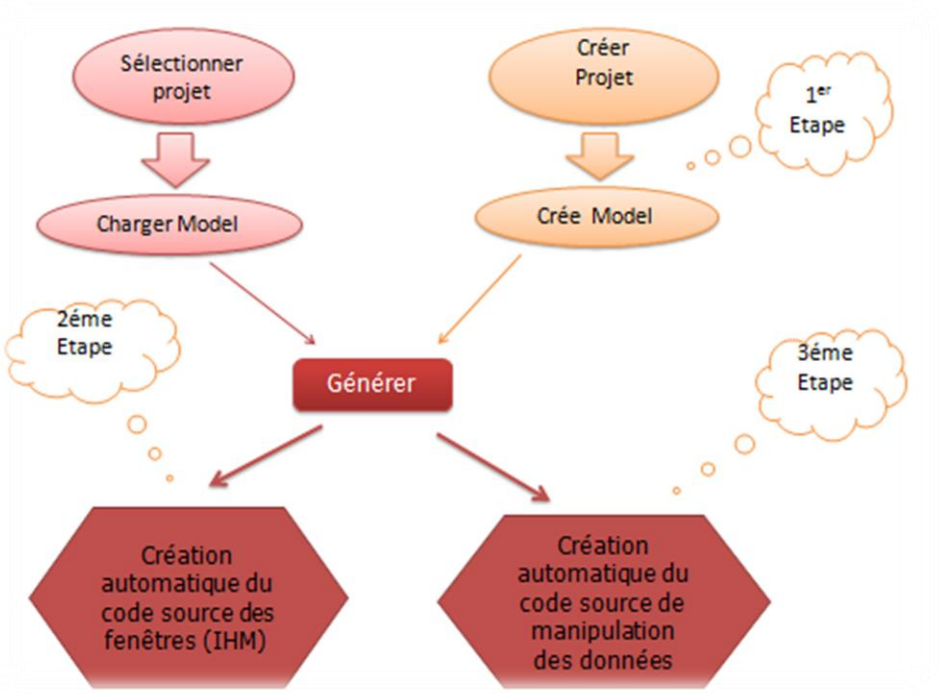


Figure 4.1 : *Représentation de notre travail*

- **Identification des registres**

Dans cette étape nous voulons identifier les concepts de base « registres » pour créer une arborescence des besoins dont le but assurer le passage de ces derniers vers un modèle permettant la génération d'un code source Java.

Les besoins sont fixés en gestion seulement.

Exemple : Gestion des étudiants :

- Contient les attributs suivants : (Matricule, Nom, Prénom, Date de naissance …etc.) ;
- Possède les opérations de base suivantes : (Ajout, Modification, Suppression …etc.).

- **Création automatique du code source des fenêtres (IHM)**

La création du code source est assurée par la traduction de la gestion en un ensemble de fenêtres.

Exemple : Gestion des étudiants :

- Les attributs vont créer une fenêtre Formulaire Etudiant avec un bouton Valider, l'IHM est créée automatiquement par une procédure de génération.

- **Création automatique du code source de manipulation des données**

La création du code source des différents boutons créer dans l'étape IHM est assurée dans cette étape.

Exemple : Gestion des étudiants :

- Le bouton Valider contient un code source créé automatiquement par une procédure de génération**.**

4. Conception

4.1 Identification des diagrammes

La conception est la dernière étape avant de passer à l'implantation du notre projet. Dans cette section, nous analyserons la modélisation UML de la nouvelle approche et nous allons identifier deux diagrammes :

● Les diagrammes de cas d'utilisations représentent un intérêt pour l'analyse des besoins métier ce qui nous permettra de démarrer l'analyse orientée objet et identifier les classes candidates. [18]

● Un diagramme de classes est une collection d'éléments de modélisations statiques (classes, paquetages...), qui montre la structure d'un modèle. Les classes sont liées entre elles par des associations. Une association permet d'exprimer une connexion sémantique bidirectionnelle entre deux classes. [18]

4.2 Conception des couches

Cette phase consiste à enrichir la description du procédé, de détails d'implémentation afin d'aboutir à une description très proche d'un programme. Nous allons modéliser toute la nouvelle approche en diagramme de cas d'utilisation, et de classes.

4.2.1 Diagramme de cas d'utilisation

Le diagramme suivant représente le cas d'utilisation de notre approche (Figure 4-2). Il décrit le Comportement du système du point de vue utilisateur. En effet, l'utilisateur va dessiner (créer) un modèle ou le charger à partir d'un projet existe déjà. Ce modèle est un ensemble de registres de deux types : « principal » et « index ». Il peut aussi faire des connexions entre deux registres de type différent dans ces registres nous pouvons effectuer les actions suivantes : ajout d'un détail au registres et la suppression de ces derniers. Dans le registre (type principal) l'utilisateur peut ajouter des détails, supprimer ou modifier.

Mais dans le registre (type index) l'utilisateur peut voir seulement une table de connexion qui représente les connexions avec ces registres. Ensuite, l'utilisateur réalise le modèle et le résultat sera généré en code source Java complet.

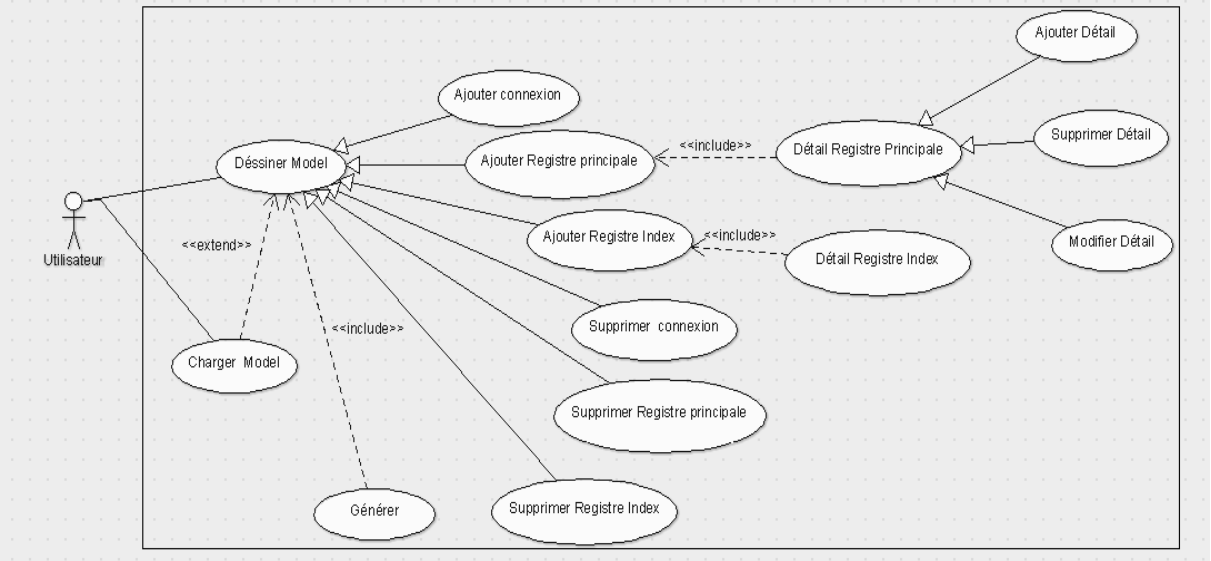


Figure 4.2 : *Diagramme de cas d'utilisation.*

4.2.2 Diagramme de classes

Le diagramme de classes représenté dans la figure 4-3 suivante décrit les associations entre les Classes et ceci afin de déterminer les dépendances entre les différentes classes.

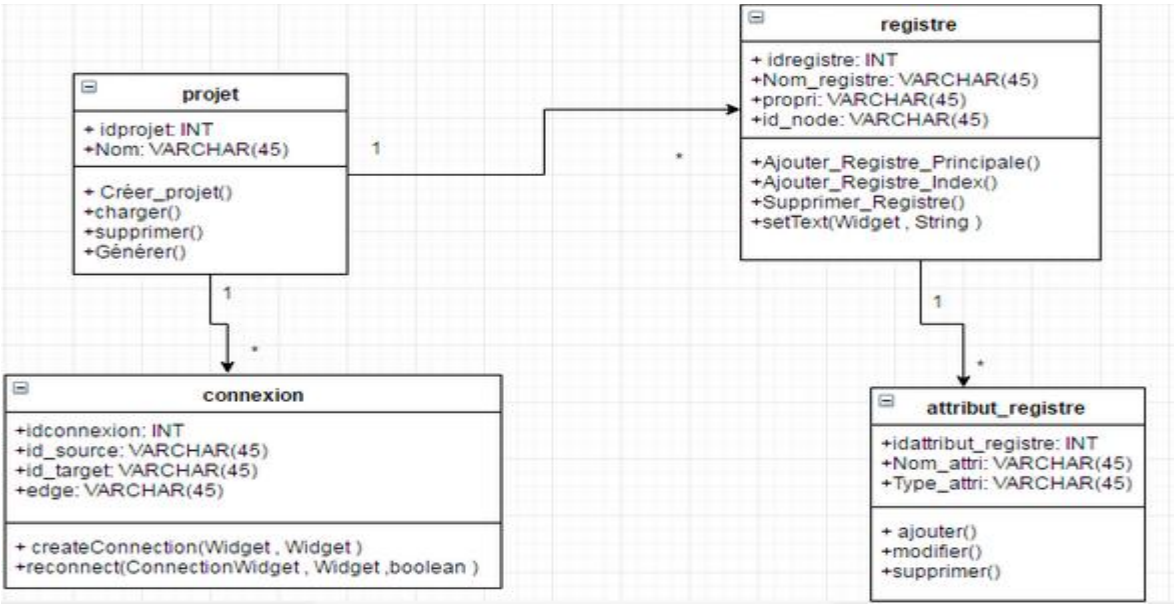


Figure 4.3 : *Diagramme de classes.*

5. Conclusion

Dans ce chapitre, nous avons identifié les diagrammes de cas d'utilisation et de classes pour faciliter la réalisation de notre prototype.

Dans le chapitre suivant nous montrerons les étapes, plus en détails, que nous avons suivies pour implémenter et réaliser notre solution.

Chapitre 5 Implémentation

1. Introduction

L'implémentation est la phase la plus importante après celle de la conception. Le choix des outils de développement influe énormément sur le coût en temps de programmation, ainsi que sur la flexibilité du produit à réaliser.

Cette phase consiste à transformer le modèle conceptuel établi précédemment en des composants logiciels formant notre système.

Dans ce chapitre, nous allons commencer par la description de l'environnement de travail puis à dégager et à élaborer les composants de notre approche.

2. Environnement de travail

L'environnement de travail est constitué par deux parties nommées environnement matériel et environnement logiciel.

2.1 Environnement matériel

Le développement de l'environnement matériel est caractérisé par :

1. Système d'exploitation : Windows 7 Édition Intégrale 32-bit (6.1, Build 7600) (7600.win7_gdr.110408-1633).
2. CPU : Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz (4 CPUs), ~2.4GHz
3. Mémoire : 4096MB RAM

2.2 Environnement logiciel

L'environnement logiciel consiste les composants suivants :

- Le langage JAVA :

❖ Nous avons utilisé le langage JAVA, Ce choix est justifié par ses nombreux avantages dont voici quelques-uns :

- ✓ Il est orienté objet : permet l'encapsulation, le polymorphisme, et l'héritage, et qui vont nous aider à bien organiser et structurer l'application.
- ✓ la portabilité : car le compilateur java produit un code intermédiaire qui sera interprété par une JVM.

- ✓ la robustesse : bonne gestion de mémoire (nettoyage par garbage collector, pas d'accès direct à la mémoire), et une bonne gestion d'exception.
- ✓ Il est distribué : API réseaux (socket), Applet, Servlet, RMI,...etc.
- ✓ Il dispose d'une bibliothèque de classes très riches.

- Outil de développement NetBeans IDE 8.1

C'est un environnement de développement intégré (EDI), En plus de Java, il permet également de supporter différents autres langages,
comme C, C++, JavaScript, XML, Groove, PHP et HTML de façon native par l'ajout de greffons. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web). [19]

- MySQL server

Nous avons utilisé MySQL server qui est système de gestion de bases de données relationnelles (SGBDR). Pour le stockage des données de notre base.

- MySQL Workbench.

MySQL Workbench (anciennement *MySQL administrator*) est un logiciel de gestion et d'administration de bases de données MySQL créé en 2004. Via une interface graphique intuitive, il permet, entre autres, de créer, modifier ou supprimer des tables, des comptes utilisateurs, et d'effectuer toutes les opérations inhérentes à la gestion d'une base de données. Pour ce faire, il doit être connecté à un serveur MySQL.[20]

3. Implémentation

Dans cette partie, nous allons présenter les différentes phases de la réalisation de notre projet en mentionnant des imprimes écrans de notre application.

3.1 Prototype

Le prototype réalisé s'identifie par une interface (Figure 5-1) où sont définies toutes les étapes principales réalisées. Un ensemble d'items est défini dans le menu bar (Fichiers) pour identifier et choisir (charger model), dessiner model et quitter le système.

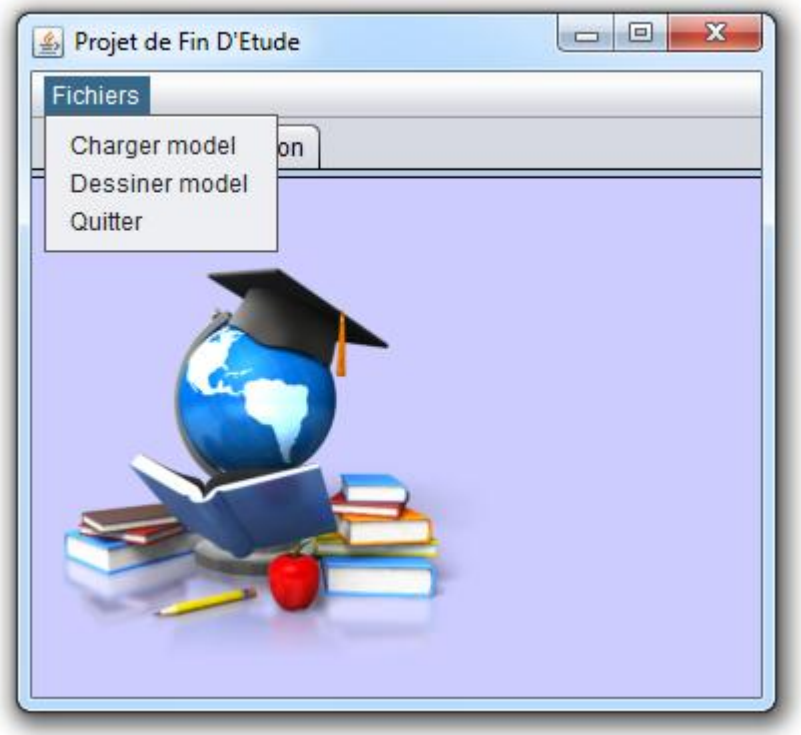


Figure 5.1 : *Interface d'accueil*

3.2 Choix Charger model

La spécificité de cette interface est la possibilité de choisir un projet déjà existant pour voir son model (en cliquant sur projet1) (Figure 5-2).

Le nom de projet est déjà enregistré dans une table **« projet »** dans une base de donnée (mybdd) (Figure 5-3).

Puis, l'interface dans la figure (Figure 5-4) représente le modèle de ce projet. Les noms des registres sont enregistrés dans une table « **registre** » qui contient : le nom de registre (par défaut), propriété (principale ou index), nouveau nom (id_node), et identificateur de projet correspondant (Figure 5-5).

- Les liens (connexions entre les registres) aussi sont enregistrés dans une table **connexion »** qui contient : le nom de registre source (par défaut), cible (par défaut), le nom de connexion et Identificateur de projet correspondant (Figure 5-6).
- Pour faire les liens il suffit de cliquer sur « ctrl » de clavier et « bouton gauche » de la sourie.

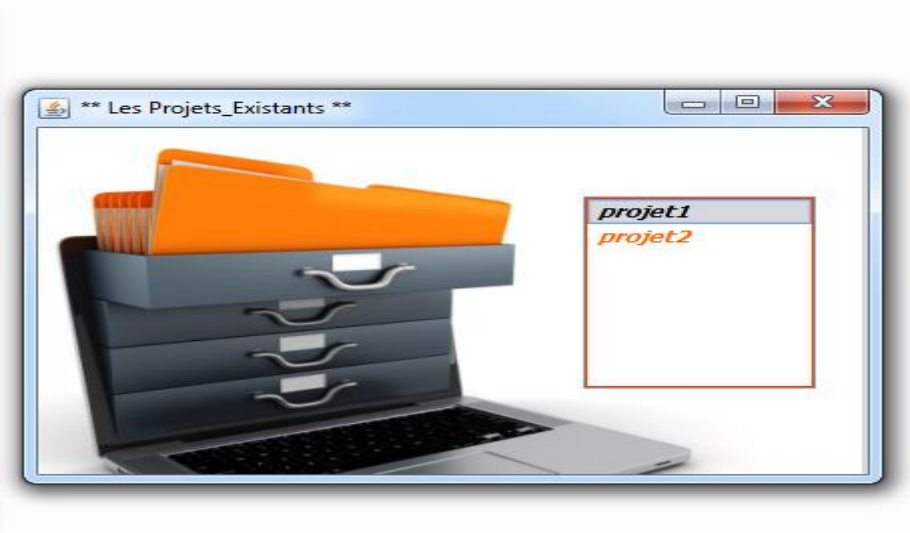


Figure 5.2 : *Choix de projet1*

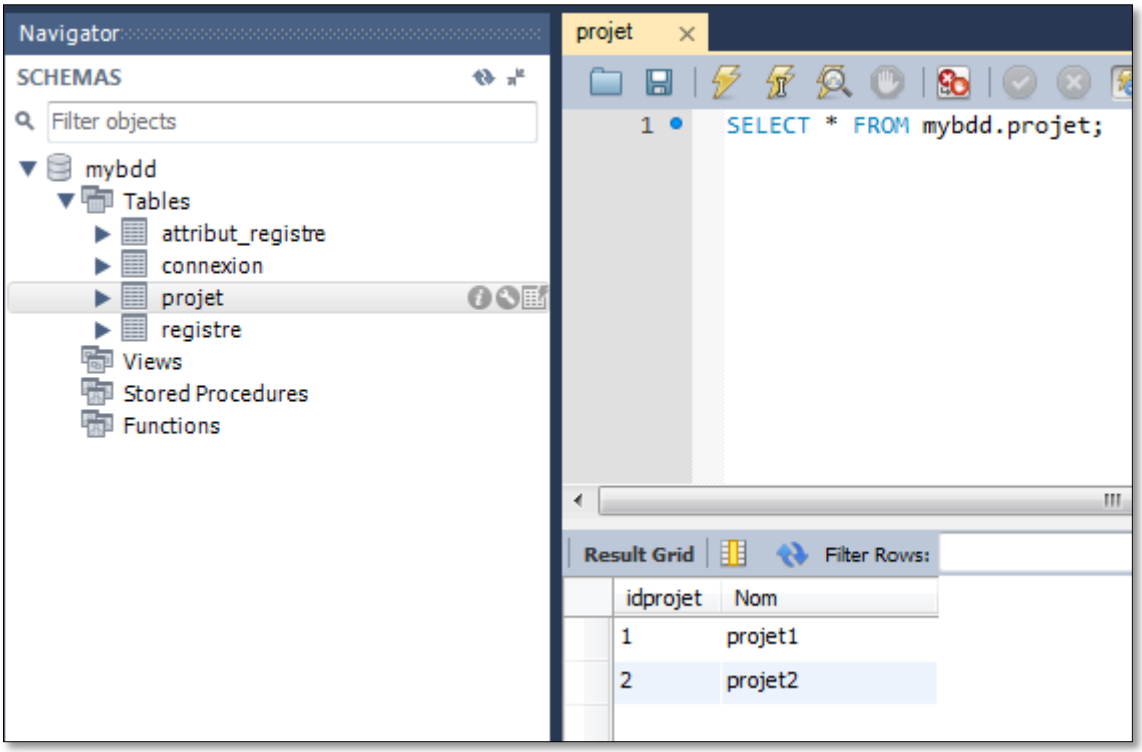


Figure 5.3 : *la table «projet»*

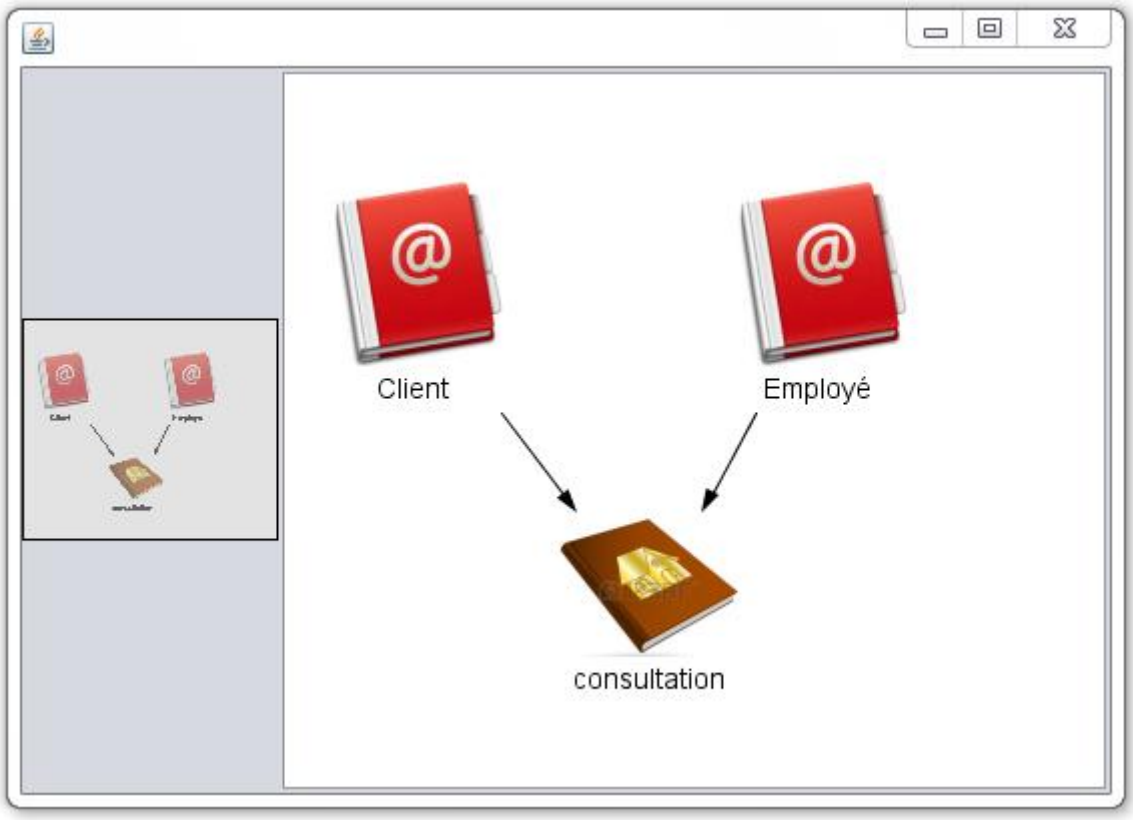


Figure 5.4 : *Model pour projet1*

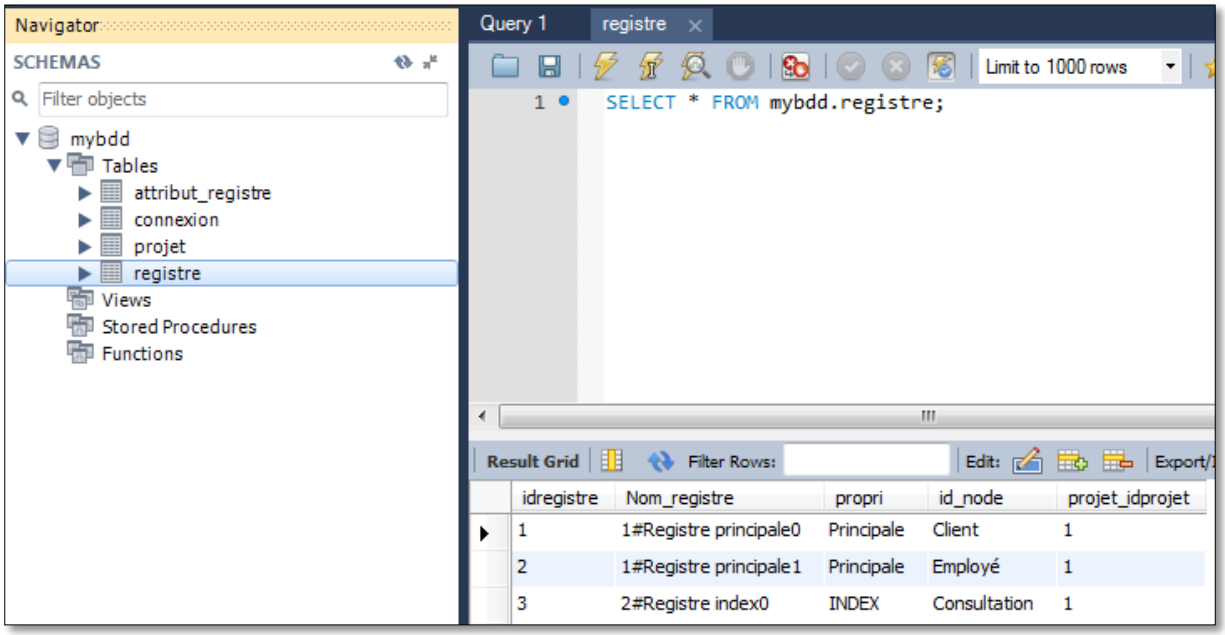


Figure 5.5 : *Table registre*

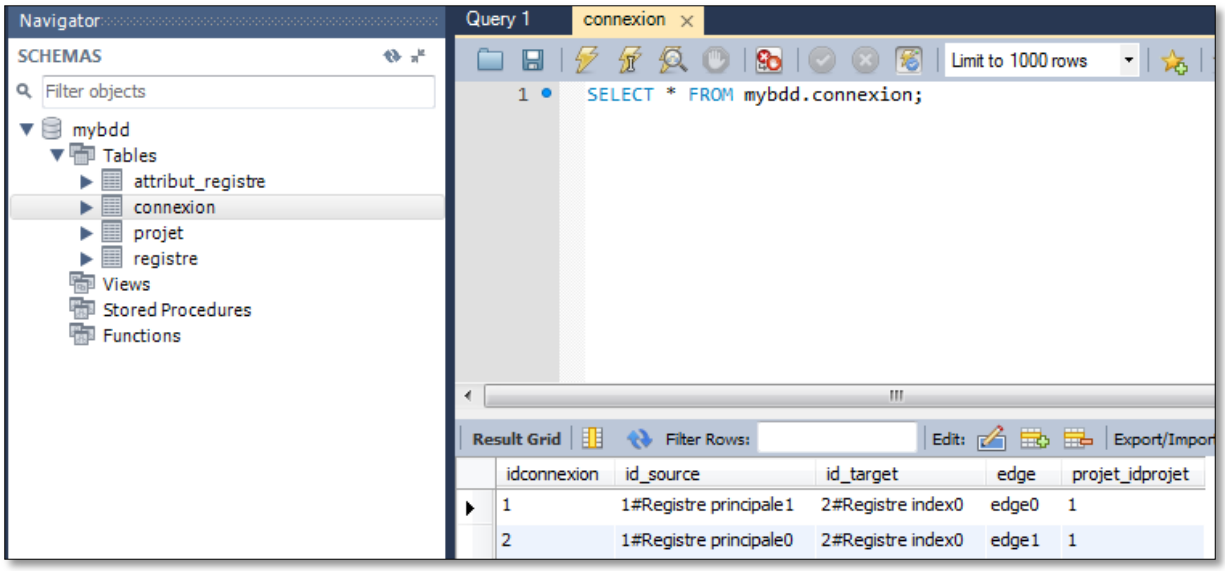


Figure 5.6 :*Table « connexion »*

Un Clic droit dans l'espace blanc permet de choisir trois actions : Ajout d'un registre de type principal ou bien de type index, et génération du code source (Figure 5-7).

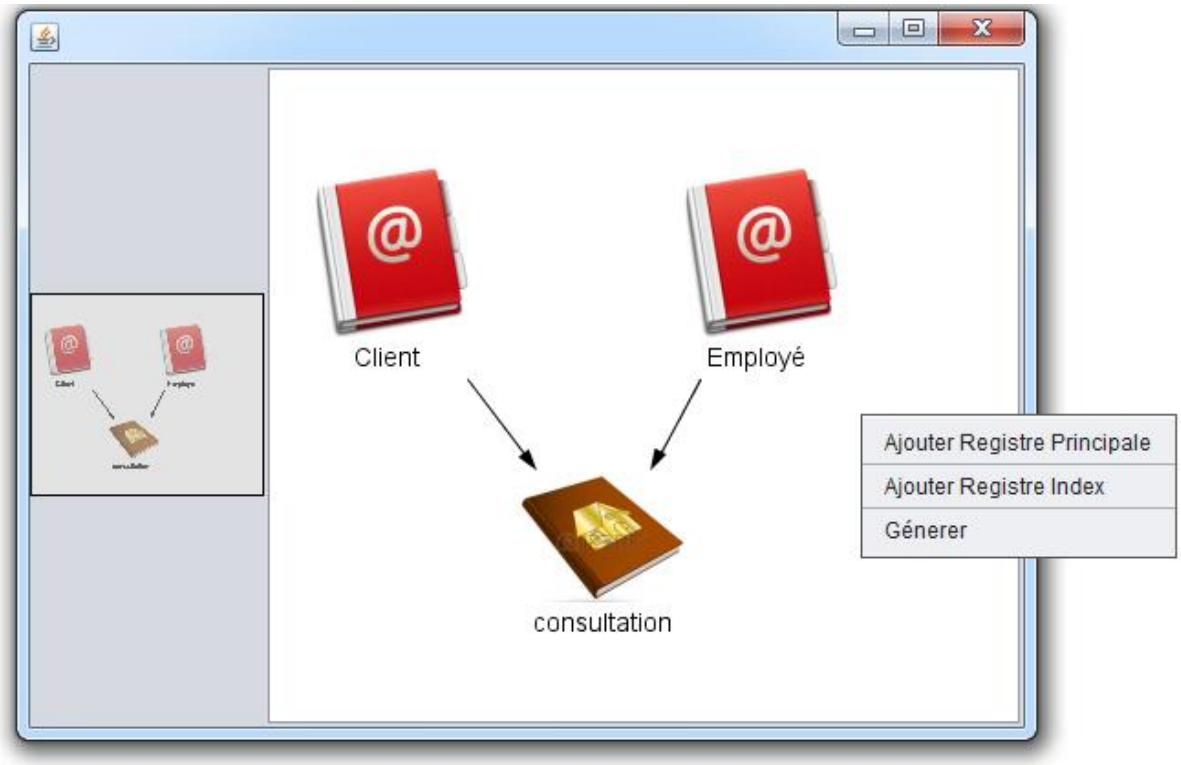


Figure 5.7 : *les trois actions sur le model*

3.2.1 Choix (Ajouter registre principale ou Ajouter registre index)

Après avoir choisir l'action « Ajouter registre principale » (Figure 5-8), nous pouvons modifier le nom de registre par défaut (1#Registre principale2) (double clic sur le nom de registre). Si le nom de registre est déjà utilisé nous allons récupérer l'ancien nom et afficher une boite de dialogue pour confirmer (Figure 5-9)

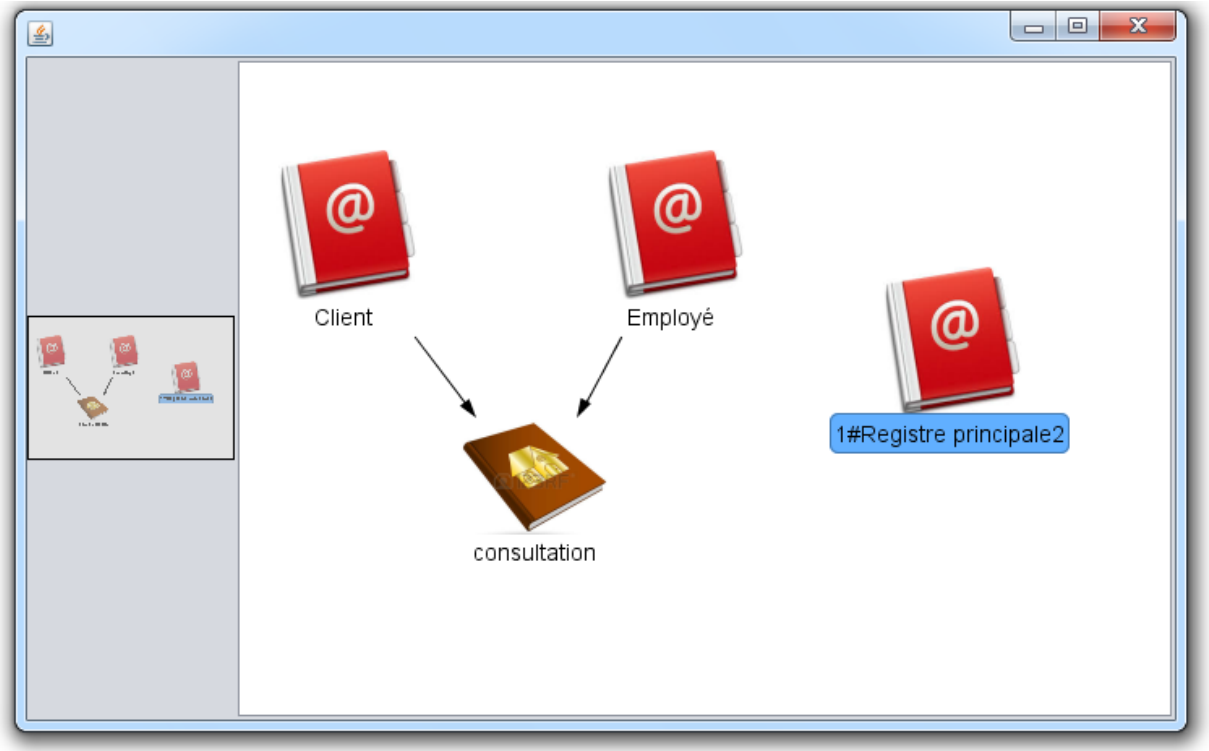


Figure 5.8 : *Ajouter un registre principale*

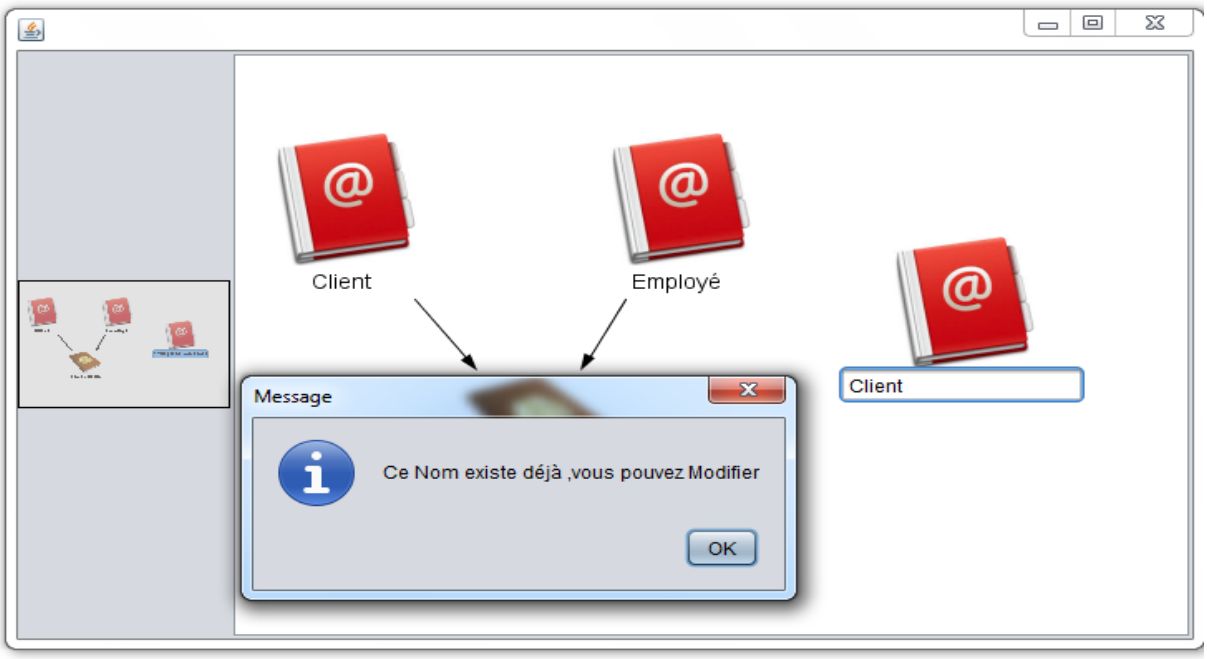


Figure 5.9 :*Contrôle pour le nom de registre*

- Nous ne pouvons pas relier le registre principal avec un autre registre principal, mais nous pouvons avec le registre index.
- Un clic droit sur n'importe quel registre va définir deux actions (Figure 5-10) : « supprimer registre » et « détaille registre ».

❖ **Choix supprimer registre**

L'action permet la suppression du registre.

❖ **Choix détaille registre (principale)**

Cette action affiche une interface contient le nom de registre avec un tableau vide contient les détails : Nom (attribut) et le type pour le remplir, sinon elle affiche un tableau contient des attributs avec les types, dans ce cas nous pouvons créer un autre détail. (Les détails de registre sont enregistrés dans une table **« attribut_registre »** (Figure 5-11))

- Si vous avez inséré les mêmes attributs existants, alors nous allons afficher une boîte de dialogue pour vous informer.

❖ **Choix détaille registre (Index).**

Cette action affiche une interface contient le nom de registre avec un tableau contient les registres qui sont liés avec lui (Figure 5-12).

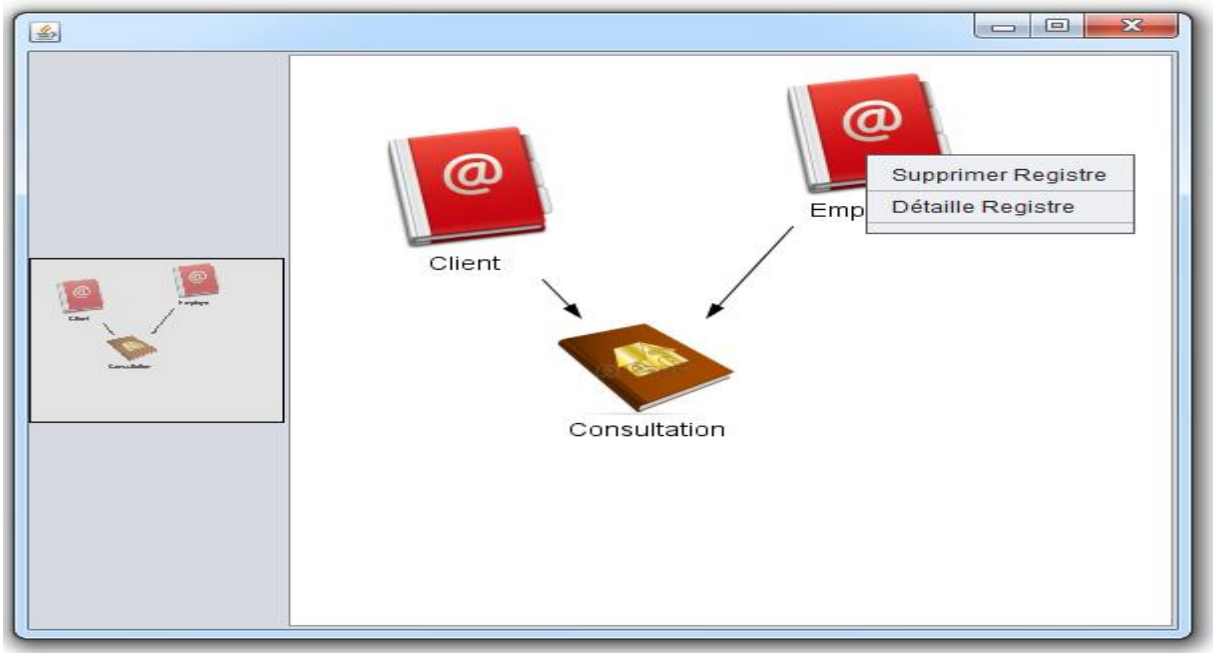


Figure 5.10 : *Action sur les registres*

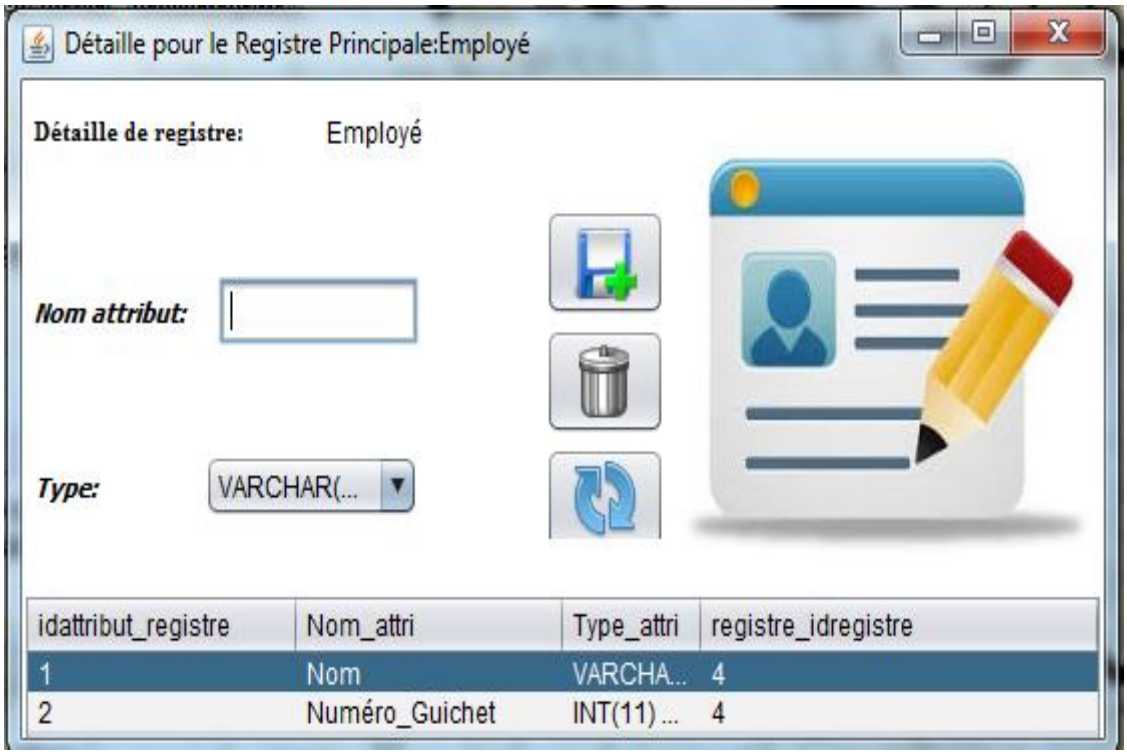


Figure 5.11 : *détail registre (principal)*

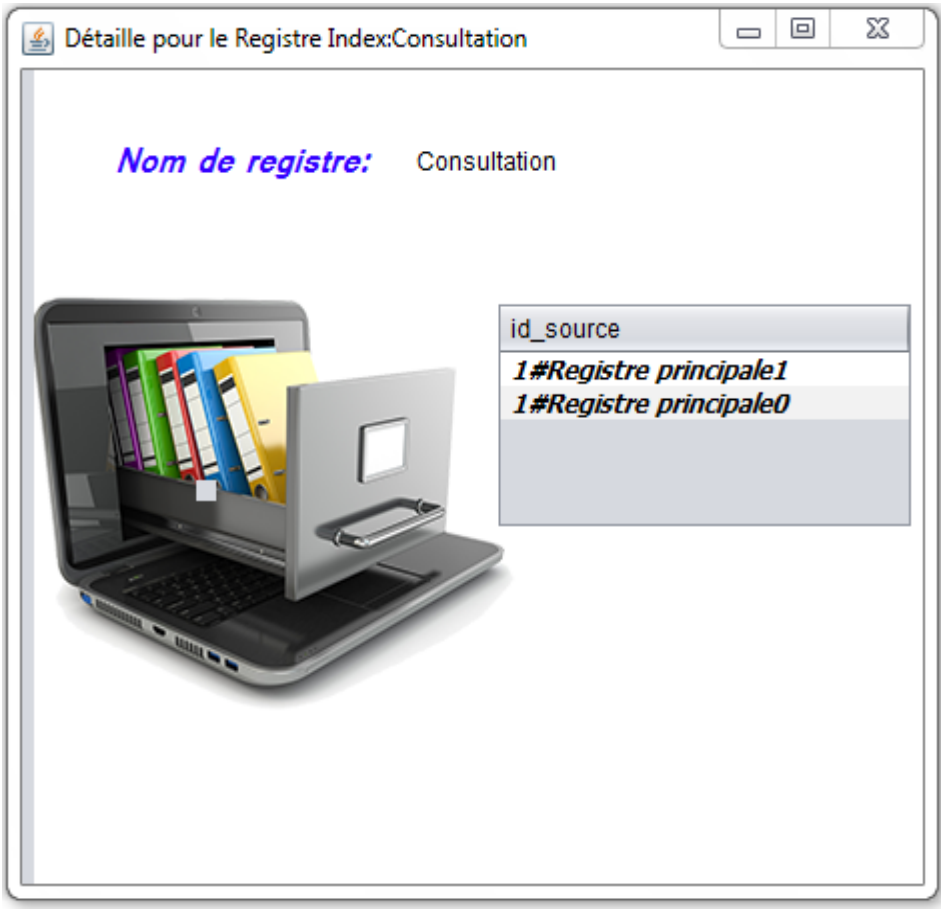


Figure 5.12 : *Détail registre (index)*

3.2.2 Choix Générer

La spécificité de cette action se résume comme suit (après un clic) :

A. **Générer automatiquement la base de données**

Permet de transformer le nom de projet vers un nom de base de données est les noms des registres de ce projet vers des tables (Figure 5.13) avec des attributs (voir Figure 5.11).

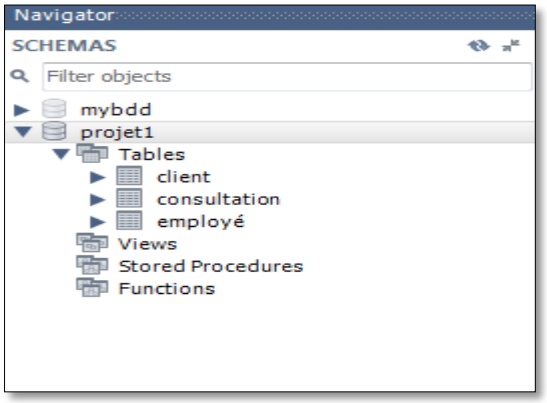


Figure 5.13 : *Génération de da base de données*

- Génération de la base de données

```
Command = "CREATE SCHEMA `"+Créer_Projets.nommodel +"`;";

conn=connextion.getConnextion();

PreparedStatement pst = conn.prepareStatement("SELECT id_node FROM
`mybdd`.`registre` WHERE  `projet_idprojet`= ? ;") ;

// pst.setString(1, propri());

pst.setString(1, recupereridmodel());

ResultSet Rs = pst.executeQuery();

while(Rs.next()) {

String   pass1 = Rs.getString(1);

attrkey = "id_"+pass1 ;
```

- La génération des tables :

❖ Pour le registre index :

```
if(prop_registre(pass1).equals("INDEX")){

      Command= Command+"\n"+"CREATE TABLE `"+Créer_Projets.nommodel
+"`.`"+pass1+"`( `"+attrkey+"` INT NOT NULL AUTO_INCREMENT ,`date` DATE
NOT NULL ";

    String noudtarget=Noudtarget(pass1);

 Vector  vec= nomSource(noudtarget);

 for(int i=0;i<vec.size();i++){

  String v1= nom_registre(vec.elementAt(i).toString());

  String pp="id_"+v1;

  String oo="VARCHAR(45)NOT NULL";
```

❖ Pour le registre principal :

```
else{     Command= Command+"\n"+"CREATE TABLE `"+ Créer_Projets.nommodel
+"`.`"+pass1+"`( `"+attrkey+"` INT NOT NULL AUTO_INCREMENT";

   PreparedStatement pst1 = conn.prepareStatement("SELECT Nom_attri
,Type_attri FROM `mybdd`.`attribut_registre` WHERE "

      + "`registre_idregistre`= ?  AND `projet_idprojet`=? ;") ;

 pst1.setString(1, affi(pass1));

 pst1.setString(2, recupereridmodel());

 ResultSet Rs1 = pst1.executeQuery();

  while(Rs1.next()) {

     String   pass2 = Rs1.getString(1);

     String   pass3 = Rs1.getString(2);

 Command= Command+",`"+pass2+"` "+pass3+ "\n";
```

- La table consultation (index) contient les identifiants des registres principaux (client et employé).

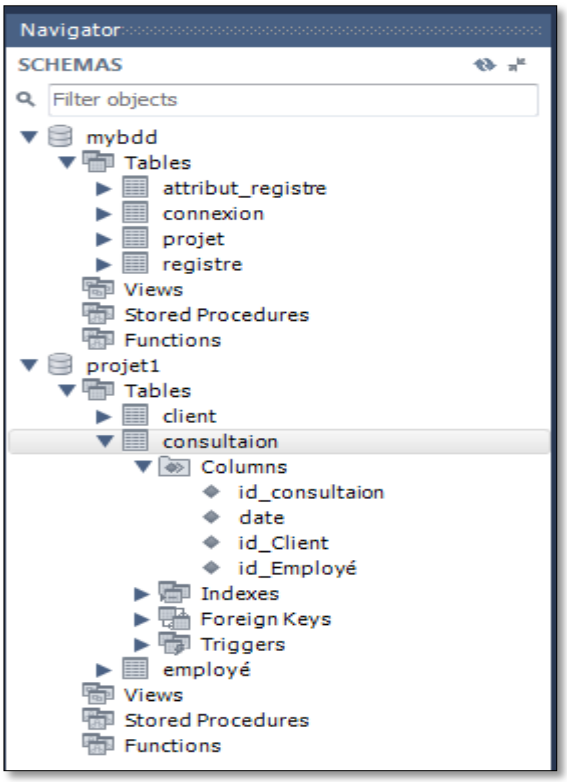


Figure 5.14 : *Le contenu de la table index*

B. **Générer automatiquement les interfaces.**

Affiche une interface qui contient les différents registres de projet (Figure 5.15) :

- Le clic sur le nom de registre principal (exemple client) permet d'afficher une table « client » qui contient les différentes informations pour les clients (Figure 5.16), dans cette table nous pouvons supprimer, modifier et ajouter les informations.

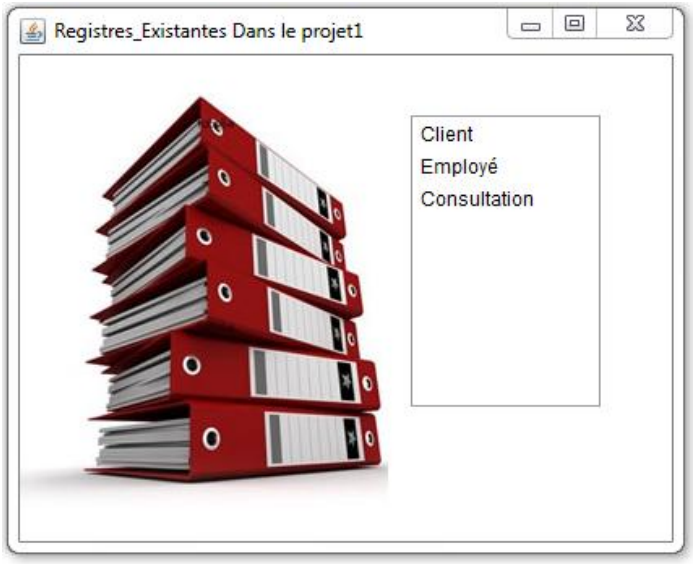


Figure 5.15 : *Liste de registre pour projet1*

- Le clic sur le bouton « Ajouter » permet d'afficher un formulaire pour le registre client de l'exemple précédent pour ajouter un nouveau client (Figure 5-17).

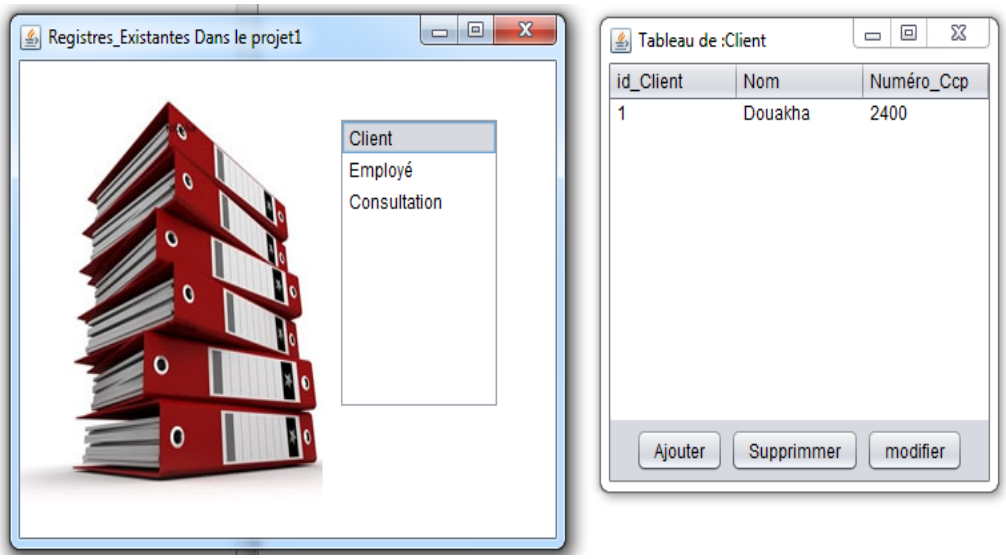


Figure 5.16 : *Table client*

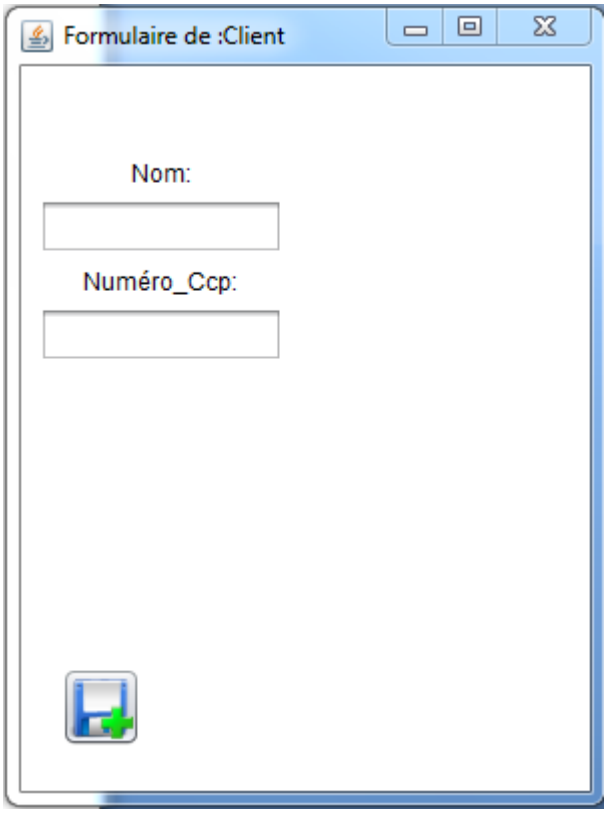


Figure 5.17 : *Formulaire pour le registre client*

- Le clic sur le nom de registre index (exemple consultation) permet d'afficher une table « consultation » qui contient les différentes informations pour les consultations (Figure 5.18).
- Dans cette table nous pouvons supprimer (Botton Supprimer), et ajouter les informations (Botton Ajouter).
  - Un clic sur le Botton Ajouter permet d'afficher une interface qui contient une date dynamique (nous pouvons la modifiée) et les déférentes informations de chaque registre sous forme de liste (Figure 5.19).

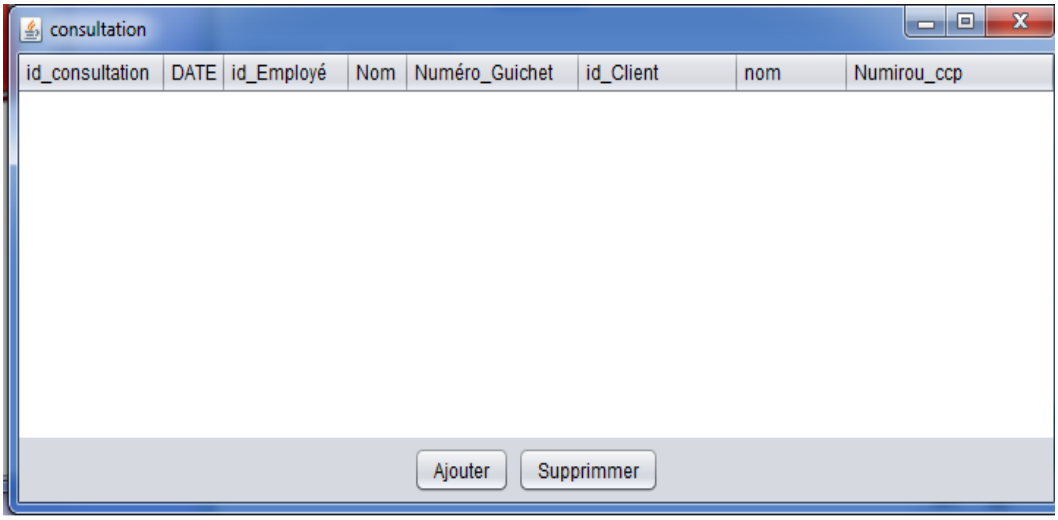


Figure 5.18 : *Tableau index pour consultation*

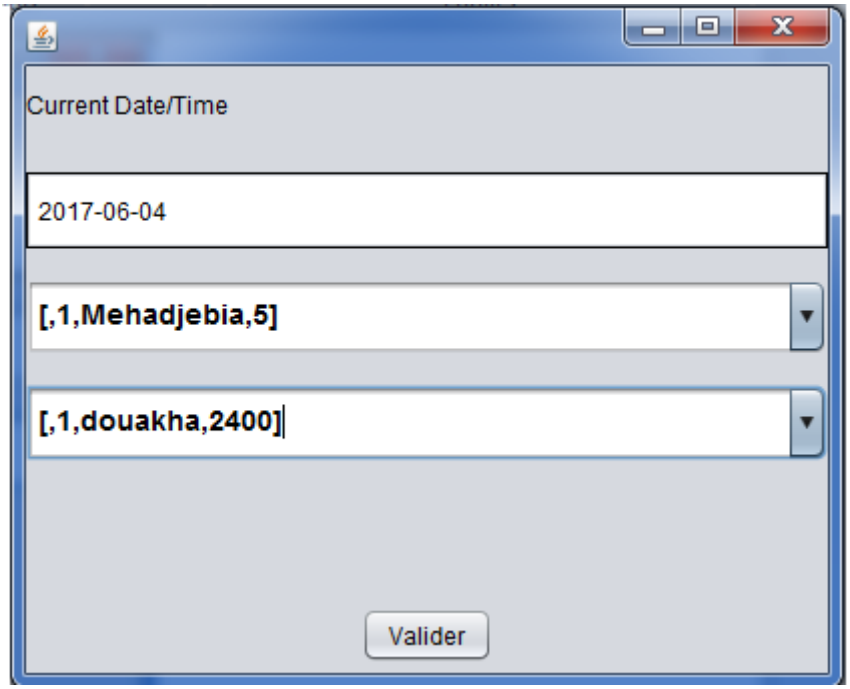


Figure 5.19 : *Formulaire pour registre index*

- Un clic sur le Bouton valider permet d'enregistrer les informations sur la table index (consultation).

C. Générer automatiquement le code source complet

Nous pouvons dans cette étape voir un code source complet utilisable de n'importe quel projet. Le code se trouve dans un dossier nommé avec le même nom de projet créé, avec les dossiers de projet principal (Figure 5.20).

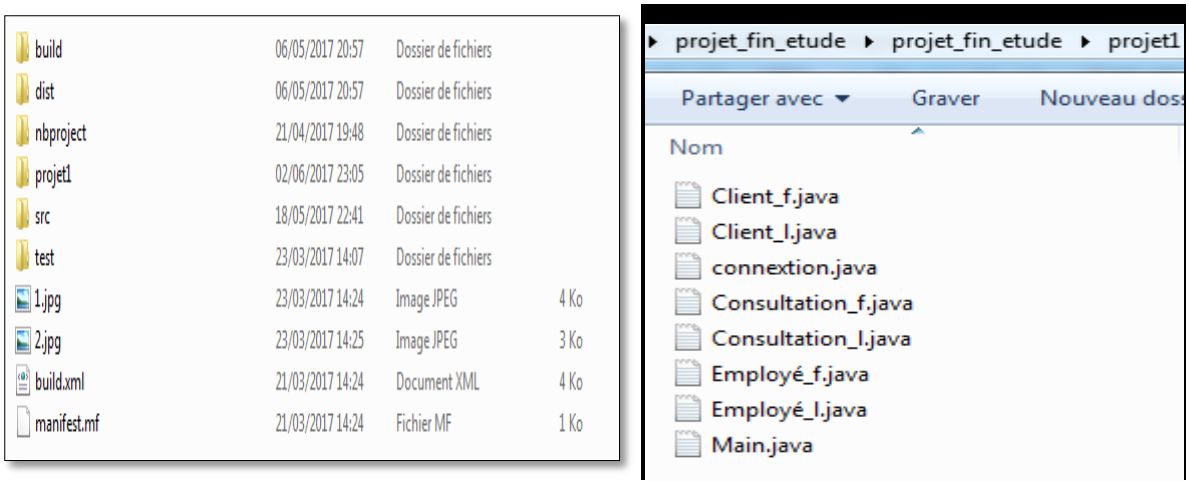


Figure 5.20 : *L'emplacement de code source généré*

Nous rajoutons la lettre « f » pour identifier un fichier « .java » d'un formulaire de chaque registre comme suit« le nom de registre_f ». Et nous rajoutons la lettre « l » pour identifier un fichier « .java » d'une liste de chaque registre comme suit « le nom de registre_l ».

Exemple : « Client_f » et « Client_l ».

La class main contient les déférents registres (Figure 5.21) ,

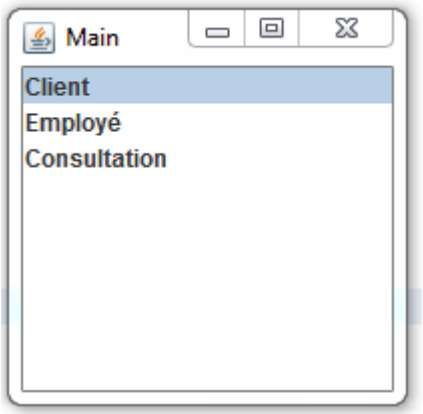


Figure 5.21 : *Class main*

3.3 Choix Dessiner model

Pour dessiner un modèle, il suffit de créer le nom de projet (Figure 5.22), ensuite vous pouvez passer à l'interface de dessin (Figure 5.23), pour dessiner le modèle vue dans la partie présenté précédemment dans le point de chargement d'un modèle (Figure 5.7).

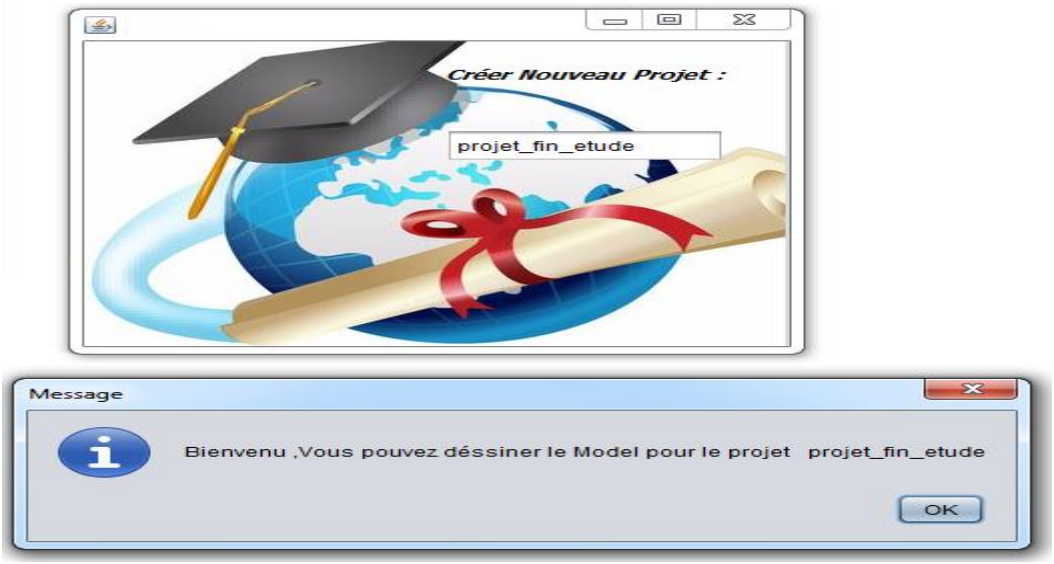


Figure 5.22 : *Interface pour créer le nom de projet*

Figure 5.23 : *Interface pour dessiner un modèle*

- Pour dessiner le modèle il suffit de cliquer sur le bouton droit de la sourie sur l'espace blanc pour afficher un menu de trois actions (Figure 5.24) et suivez les étapes suivantes :

Figure 5.24 : *Actions pour dessiner le modèle*

A. Ajouter des registres principaux (au minimum deux registres)

- Les noms des registres par défaut (1#Registre principale0, 1#Registre principale1…). Pour modifier le nom de chaque registre nous cliquons sur le nom puis le renommer.
- Nous devons ajouter les attributs de chaque registre par clic droit de la sourie sur le registre et choisir l'action" détaille registre" (figure 5.25).

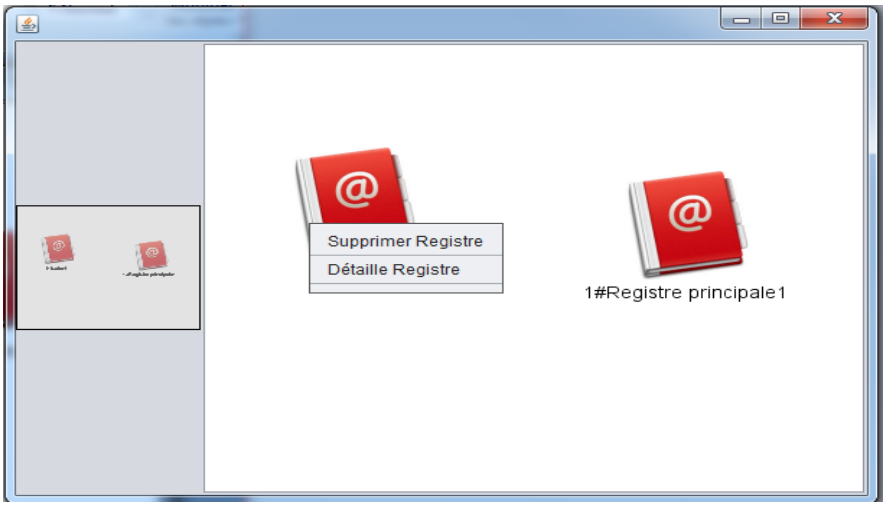


Figure 5.25 : *L'action « détaille registre »*

B. Ajouter des registres index

La spécificité de ces registres est la possibilité de faire des relations entre des registres principaux avec la caractéristique concernant la date.

- Les noms des registres par défaut (2#Registre index0, 2#Registre index1…) Pour modifier le nom de chaque registre nous cliquons sur le nom puis le renommer.
- Nous pouvons voir les détails de ces registres qui contiennent les registres principaux reliés avec eux par un clic droit sur le registre.

C. Faire des liens (connexion)

- Pour faire les liens (flèches) entre le registre index et les registres principaux, il suffit de cliquer sur « ctrl » du clavier et le « bouton gauche » de la sourie. voir l'exemple précédent dans la figure (Figure 5.4).

4. Conclusion

Dans ce chapitre, nous avons présenté le prototype réalisé. Ce prototype a réussi d'avoir un avantage de plus par rapport à d'autres outils existants pour le but de réaliser le même but.

De plus, il représente une vraie convivialité d'utilisation du fait de pouvoir choisir, charger un modèle ou dessiner ce dernier, ainsi de générer automatiquement le code source Java complet.

Conclusion générale

Nous sommes arrivés maintenant au terme de ce mémoire. Celui-ci visait concrètement à implémenter un générateur de code automatique à partir d'un dessin. Nous avons pu démontrer que ceci est possible notamment grâce à l'outil « Visual netbeans ». Ce qui nous permet de faire le dessin.

Cette génération passe toutefois par quelques étapes. En effet, nous avons d'abord sauvegardé dans une BDD les données représentées dans le modèle dessiné par l'utilisateur. Ensuite, l'outil nous permet de générer une BDD automatiquement, cette nouvelle BDD intervient dans la création automatique de code source des fenêtres (IHM), ainsi le code source de manipulation des données. Enfin, permet de créer des fichiers Java, c'est-à-dire des classes Java.

Perspectives

Le premier but du travail a été atteint. Toutefois, ce générateur est ouvert à de futurs travaux visant l'amélioration de celui-ci. En effet, nous pourrions par exemple élargir notre domaine d'application vers d'autre problème plus complexes pas seulement les problèmes de gestion « comme les sites web, les applications mobiles … ».

Bibliographie

[1]	http://www.memoireonline.com/05/12/5885/m_Generation-automatique-du-code-java--partir-d-un-modele-de-classe-UML19.html
[2]	James Phillips «Automatic Code Generation»
[3]	https://www.techopedia.com/definition/17062/code-generator
[4]	Automatic Code Generation_Embedded Control Systems _Fall 2012
[5]	Bizagi BPM Suite_Functional Description
[6]	http://younessbazhar.developpez.com/eclipse/introacceleo
[7]	https://fr.wikipedia.org/wiki/G%C3%A9n%C3%A9rateur_automatique_de_programmes
[8]	http://www.automatesintelligents.com/echanges/2002/dec/guedj.html
[9]	http://marcautran.developpez.com/tutoriels/eclipse/uml-designer/
[10]	http://laine.developpez.com/tutoriels/alm/mda/generalites-approche-mda/#
[11]	L'architecture dirigée par les modèles (MDA) -F.-Y. Villemin, CNAM .
[12]	Université de Pau et des Pays de l'Adour «Ingénierie des Modèles Transformations de Modèles-Eric Cariou » Département Informatique
[13]	Examen probatoire du diplôme d'ingénieur C.N.A.M.en INFORMATIQUE-option ingénierie et intégration informatique : système de conduite
[14]	http://www.guideinformatique.com/dossiers-actualites-informatiques-developpement-e-commerce-15/generation-automatique-dapplications-55.html
[15]	https://doc.pcsoft.fr/?2031042&name=creer_une_application_grace_rad
[16]	http://www.experts-tourisme.fr/developpement-applications-s412312.htm
[17]	http://istqbexamcertification.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it
[18]	Sana SELLAMI « Conception et Réalisation d'un outil de génération automatique de Mappage pour la transformation de documents XML»
[19]	https://fr.wikipedia.org/wiki/NetBeans Dernière modification de cette page le 6 avril 2017, à 20:25.
[20]	https://fr.wikipedia.org/wiki/MySQL_Workbench Dernière modification de cette page le 30 janvier 2017, à 10:39.