PROJET DE SEMESTRE

PROGRAMMATION ORIENTÉE OBJET - JAVA

WISSAD Sara et ZAKI Oussama - Encadré par Mr. A. NEJEOUI

L'objectif principal du projet est de mettre en œuvre les notions de bases de la programmation orienté objet - POO, en utilisant le langage JAVA, ainsi que de s'initier à la gestion de projet et au développement d'un logiciel.

TABLE OF CONTENTS

REMERCIEMENT

De prime abord, nous tenons à exprimer notre gratitude et notre reconnaissance sincère à Mr. NEJEOUI Abderazzak, Professeur du module de Programmation orienté objet JAVA à l'Ecole Nationale des Sciences Appliquées de Marrakech. Auprès de lui, nous avons reçu une aide constante, éclairée et stimulante, ses conseils riches d'enseignements et ses encouragements ont été pour nous des apports déterminants dans la réalisation de ce travail.

Nous nous souviendrons toujours de : « «*Un bon ingénieur* n'est pas synonyme de majorant durant les études, on trouve souvent des ingénieurs avec des bonnes notes mais qui sont nuls, c'est en travaillant qu'on gagne de l'expérience » - Mr. NEJEOUI.

Permettez-nous d'adresser nos salutations et nos remerciements sincères à nos camarades qui nous ont prêté assistance au cours de nos études et dans la réalisation de ce projet.

PARTIE I :
PRESENTATION DU PROJET

L’objectif de cette partie est de donner une idée générale sur le projet, et d’établir un plan à sa conception selon le cahier de charge prédéfinis

1- PRÉSENTATION DU PROJET

Etant donné l'importance que nous accordons à l'achat des fournitures scolaire, une facilité de gestion des librairies pour une meilleure exploitation s'impose d'où la nécessité de leur proposer une solution plus optimale, moins couteuse, plus performante, et plus prémunie de tout ce qui est erreur. Une solution intégrante, utilisant des outils pour permettre la gestion de la caisse au complet.

Les libraires ont besoin d'un outil permettant un suivi rigoureux du stock, de la caisse des produits…, s'adapter aux exigences du marché et de la clientèle de manière efficace. C'est là qu'intervient notre projet dont le but est de créer une application Java Desktop de gestion de point de vente d'une librairie. Notre logiciel intitulé ***LeaderPapeterie*** saura répondre à tous les besoins listé ci-dessus, puisqu'il va assurer :

- La gestion de caisse (impression facture, retenu du montant, remise du change…).
- La gestion des articles, du stock, des fournisseurs…

Pour ainsi pouvoir gérer sa libraire en toute tranquillité tout en étant informé de toutes les notifications.

2- LOGO DE L'APPLICATION

FIGURE 1 - LOGO DE L'APPLICATION

3- CAHIER DE CHARGE

Ce projet consiste à réaliser un logiciel de gestion de papeterie. Le logiciel sera développé pour assurer 2 principales fonctions :

- Une interface permettant aux individus n'ayant aucune connaissance informatique de tenir une caisse.
- Une fonction secondaire - Une interface fournissant au gérant un système de gestion totale de la papeterie (articles, fournisseurs, ventes, achats, stock…).

4- PLANIFICATION DU PROJET

Une étape importante et qui constitue le pas de départ est la planification, qui procure une vue global sur les étapes menant à la réalisation du projet et ainsi savoir se situer au niveau de chaque étape. Ce plan consiste en deux étapes.

4-1 ETUDE DE L'EXISTANT :

Consiste en u briefing sur les logiciels de caisses disponibles dans le marché, ainsi qu' d'avis de quelques libraires. Apres une étude du marché, nous avons pu établir une liste des logiciels utilisés : Open Concerto, Hiboutike, Cyber Plus, EBP…

- ***Problématique***

Tous les logiciels cités ci-dessus présentent des contraintes d'utilisations dont souffrent la majorité des libraires : Difficulté de manipulation, latence, incompatibilité avec un OS, design pas attirant…

- ***Solution avec Leader Papeterie :***

Les modifications apportées se résument aux points suivants :

- Un logiciel simple, convivial, ergonomique et facile
- Un logiciel qui se résume juste aux besoins demandés ; éliminer les fonctionnalités supplémentaires.
- Le design de l'interface doit être adapté à une utilisation quotidienne et peut être personnalisé.
- Une rapidité et efficacité lors des traitements des actions.
- Un service de maintenance appréciable.

4-2 SPECIFICATION DES BESOINS :

Une formulation des besoins à partir des informations rassemblées de l'étape précédente. L'analyse de ces informations nous a permis de spécifier les fonctionnalités de notre logiciel.

- ***Gestion Point de vente***

Notre logiciel et tout d'abord une application de point de vente, ou le cassier peut facilement construire, encaisser et enregistrer ses tickets/factures, calculer les différents prix et manipuler les différentes catégories des articles.

Pour effectuer une vente, le caissier choisi pour chaque article sa quantité et la voit affichée. Le total de la facture du même article (saisi n'importe quand) dans une même ligne pour mieux optimiser le ticket imprimé. Une fois les factures(ou tickets) rassemblées, et vers la fin de chaque mois, peut consulter des statistiques sur des produits les plus rentables, le plus vendu, les moins vendus, etc...

- ***Gestion des articles***

Notre logiciel gère les produits existants dans la librairie (tous types d'articles classés en famille pour une gestion facile et optimale). L'administrateur peut ajouter, supprimer rechercher ou modifier des articles existants.

- ***Gestion des accès et sécurité***

L'accès à la base nécessite un login et un mot de passe, de plus, le logiciel assure 2 niveaux d'accès :

- Libraire administrateur

Le propriétaire ou le responsable de la papeterie, il a le droit de réaliser toute fonction proposée par le logiciel.

- Caissier :

A des droits restreints comparé à l'administrateur, peut seulement chercher dans la base de données, ou manipuler la caisse, pour garer un niveau de sécurité propre.

- ***Gestion de la rupture de stock :***

Pour remédier les problèmes relatifs à la rupture soudaine de stock, et ne pas compromettre le déroulement des ventes, une fois la quantité de stock atteint une valeur de seuil, l'administrateur en sera avisé pour remplir son stock à nouveau.

- ***Gestion des fournisseurs :***

Pour établir une liste de tous les fournisseurs des différents produits, et ainsi pouvoir la gérer, y ajouter, supprimer, modifier en cas de changement de contacts …

PARTIE II :
CONCEPTION DU PROJET

Dans le but d'une conception complète sans oublie ou manque, différents outils bibliothèques, et framework ont été utilisés : Netbeans, Hibernate, UML, Use Case diagramme…

1- BESOINS TECHNIQUES

Pour mettre à jour un tel projet, plusieurs outils ont été mis en œuvres, tel que l'IDE, serveur de base de données, les APIs…

1-1 NETBEANS

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plateforme.

L'IDE Netbeans s'enrichit à l'aide de plugins.

1-2 JGOODIES LOOK AND FEEL :

Pluggable Look and Feel est un mécanisme utilisé dans le Java swing Widget Toolkit permettant de changer le look and feel de l' interface utilisateur graphique à l'exécution.

Swing permet à une application de se spécialiser le look and feel de widgets en modifiant la valeur par défaut (via les paramètres d'exécution), dérivant d'un existant, en créant à partir de zéro, ou, en commençant par J2SE 5.0, en utilisant le skinnable synthé look and feel, qui est configuré avec un XML fichier de propriétés. Le regard et la sensation peuvent être modifiés lors de l'exécution.

Crochets de Swing widgets d'interagir avec leur look and feel. Chaque widget défini in Swing peut déléguer sa «peinture» à ses associés l'interface utilisateur des classes appelées délégués de l'assurance-chômage.

Une API pour gérer les définitions look-and-feel existants. Cette API permet de définir l'apparence et la sensation ou changer de look et se sent à l'exécution.

1-3 HIBERNATE :

Hibernate est un Framework open source gérant la persistance des objets en base de données relationnelle. Hibernate est adaptable en termes d'architecture, il peut donc être utilisé aussi bien dans un développement client lourd, que dans un environnement web léger de type Apache Tomcat ou dans un environnement J2EE complet : WebSphere, JBoss Application Server et Oracle WebLogic Server.

Hibernate apporte une solution aux problèmes d'adaptation entre le paradigme objet et les SGBD en remplaçant les accès à la base de données par des appels à des méthodes objet de haut niveau.

- ***Modules d'Hibernate :***

Hibernate se compose de plusieurs modules développés par des équipes différentes.

- **Core**

Le module principal d'Hibernate contient les fonctionnalités clef (principalement connues depuis la version 2 de la bibliothèque) telles que les sessions, les transactions, le cache d'objet ou le langage SQL.

- **Annotations**

Apporte le support des Annotations tel que décrit dans JSR 175. Cette approche permet d'éviter la description de la correspondance entre les champs d'une table et les champs du POJO en XML.

- **Entity manager**

Permet le support de JSR 220 JPA par le module Core

- **Shards**

Ce module permet la partition horizontale du Core Hibernate

- **Validator**

Module de validation des contraintes d'entité de la base de données implanté sous forme d'annotations tel que les plages de valeurs autorisées, les formats de chaine de caractère, la détection des valeurs nulles, etc...

- **Search**

Le dernier module apporte une couche d'abstraction pour la recherche de Lucene appliquée sur les entités persistantes maintenues par Hibernate.

- **Tools**

Ensemble d'outils pour NetBeans ou Eclipse facilitant le développement avec Hibernate.

- ***Comparatif entre Hibernate et Java Data Object***

Hibernate est à la base un logiciel open-source alors que Java Data Objects (JDO) est un standard. Cependant Hibernate a inspiré le groupe de travail qui a développé la norme JPA 1.0 qui est la partie persistance de données de la norme EJB3 (JSR 220). Le serveur d'application JBoss utilise entre autres Hibernate pour sa persistance et son implémentation des EJB. La majeure partie des implémentations JDO sont compatibles JPA. La spécification JDO ne se limite pas seulement aux bases de données relationnelles ; en particulier, elle gère la persistance dans des bases de données objets ou dans des fichiers XML, ci-dessous notre diagramme.

Hibernate le permet aussi depuis sa version. Il existe des implémentations JDO supportant également l'accès à des sources mainframe, des bases de données orientées objet, Java connector architecture (JCA), Java messaging service (JMS) et également aux services (Service Web, COBOL).

Hibernate propose HQL qui est inspiré du SQL mais qui intègre également la prise en compte d'aspect objet comme les attributs des entités et le polymorphisme. JDO propose un langage de requête, JDOQL qui est inspiré de la syntaxe objet Java.

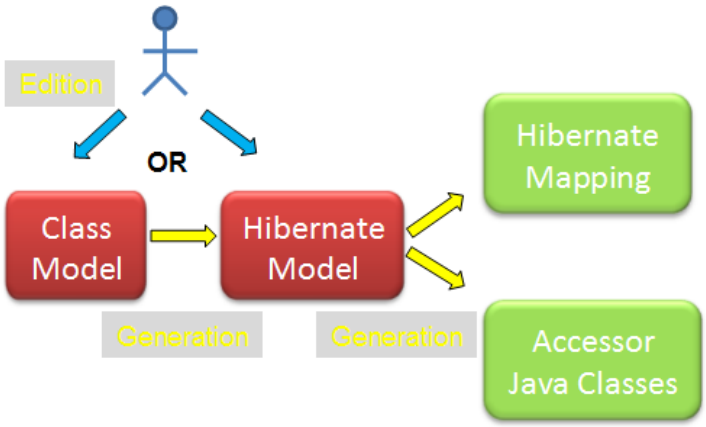

FIGURE 2 - LE MODULE MODELIO HIBERNATE DESIGNER

1-4 DIAGRAMME ENTITE ASSOCIATION

Le modèle entité-association, ou diagramme entité-association est un modèle de données ou diagramme pour des descriptions de haut niveau de modèles conceptuels de données. Il fournit une description graphique pour représenter de tels modèles de données sous la forme de diagrammes contenant des entités et des associations.

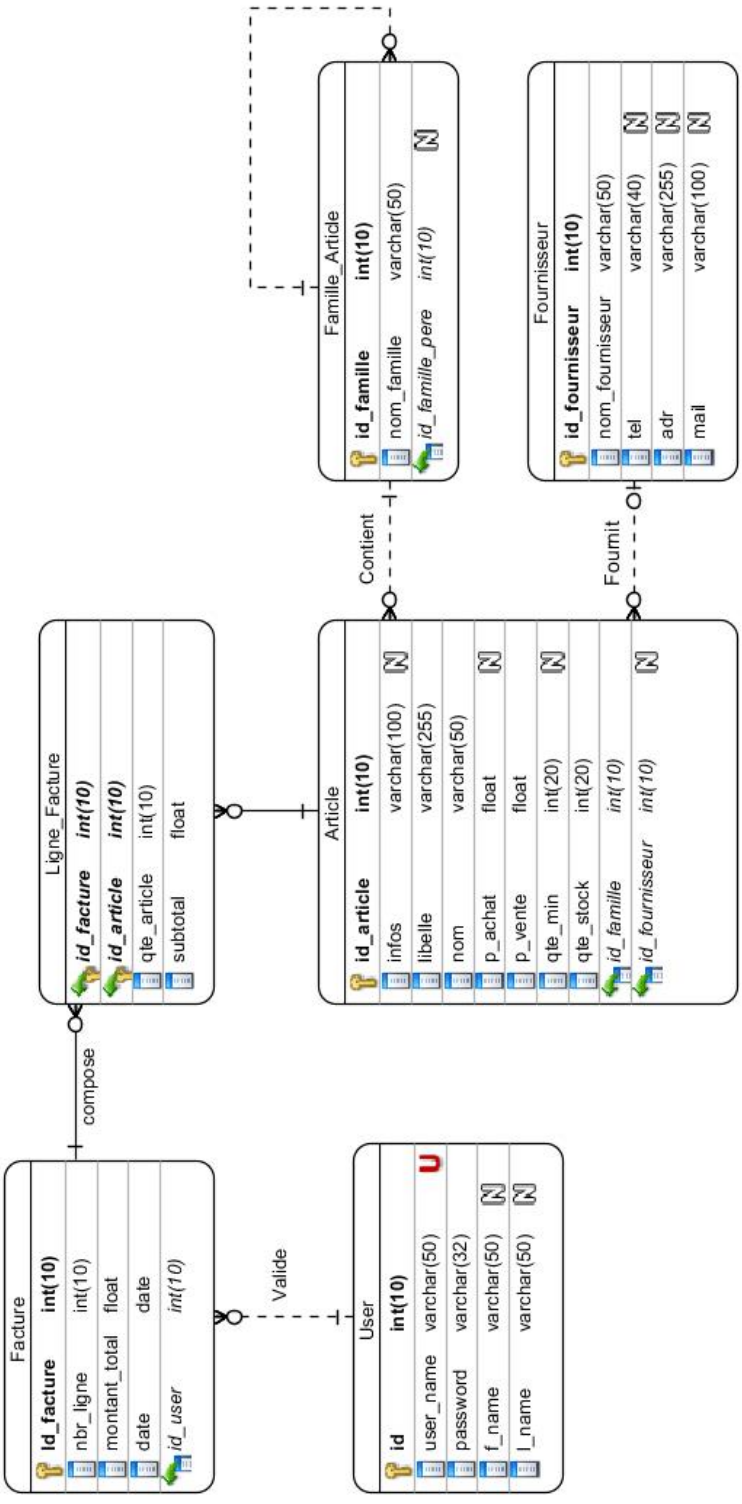

FIGURE 3 - MODELE ENTITE-ASSOCIATION

1-5 DIAGRAMME DES CAS D'UTILISATION

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel, ci-dessous notre diagramme.

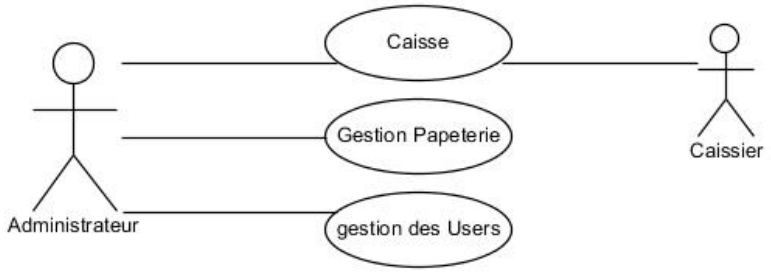

FIGURE 4 - DIAGRAMME DES CAS D'UTILISATION

PARTIE III : IMPLEMENTATION ET TEST

Cette partie comporte le développement de l'application, les différentes interfaces, implémentations et traitements.

1- CREATION DE LA CONNEXION

Sous Netbeans on crée la connexion à notre base de donnée, via les étapes illustré dans les figures suivantes.

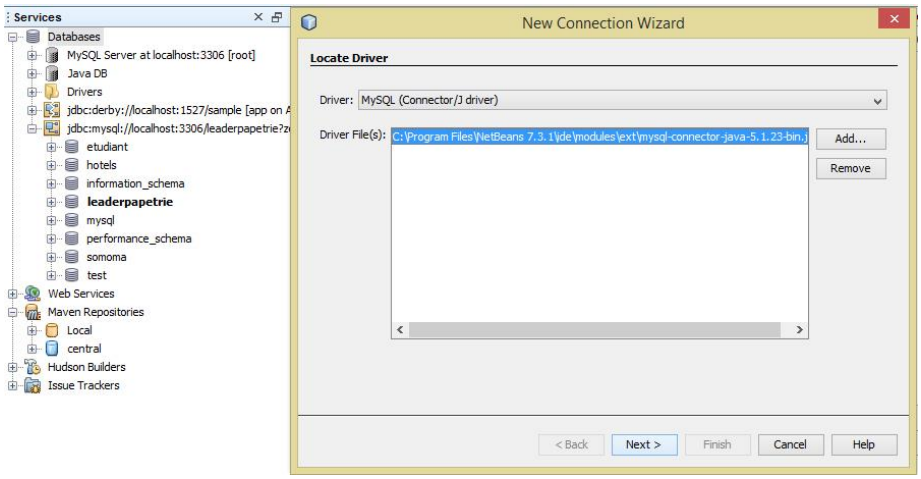

FIGURE 5 - CHOISIR LE DRIVER

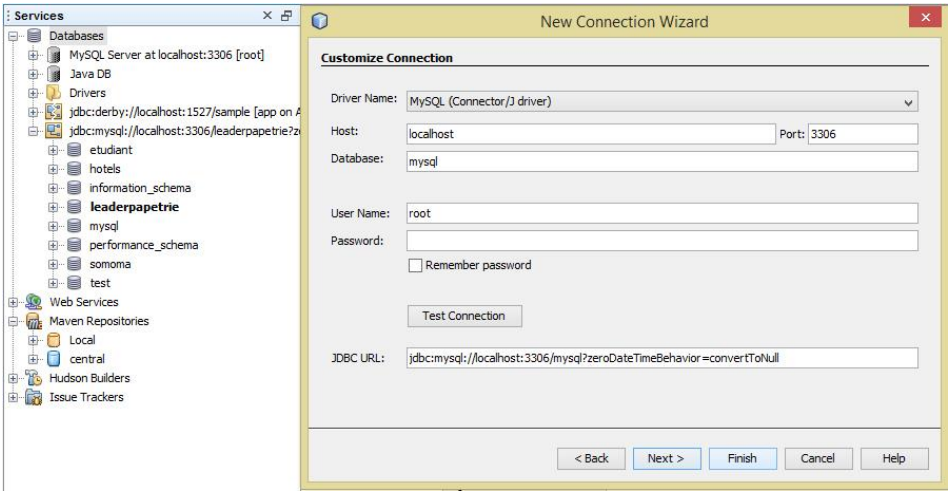

FIGURE 6 - DEFINIR LES INFORMATION RELATIVES A LA CONNEXION

2- CREATION DE LA BASE DONNEE

En s'appuyant sur le diagramme entité-association (figure3), on a pu générer le script SQL permettant la création des différentes tables à l'aide de l'outil *Visual Paradigm,* ci-dessous l'ensemble des scripts générés.

```
CREATE TABLE User(
     id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
     user_name VARCHAR(40) NOT NULL,
     psswd CHAR (40) NOT NULL,
     isAdmin TINYINT(1) DEFAULT NULL,
     f_name VARCHAR(50),
     l_name VARCHAR(50),
     PRIMARY KEY (id),
     UNIQUE KEY (user_name),
     KEY (user_name, psswd)
);


CREATE TABLE Fournisseur (
  id_fournisseur  SMALLINT NOT NULL AUTO_INCREMENT,
  nom_fournisseur VARCHAR(50) NOT NULL,
  tel             VARCHAR(40),
  adr             VARCHAR(255),
  mail            VARCHAR(100),
  PRIMARY KEY (id_fournisseur)
);


CREATE TABLE Famille_Article (
  id_famille           SMALLINT  UNSIGNED  NOT  NULL
AUTO_INCREMENT,
  nom_famille     VARCHAR(50) NOT NULL,
  id_famille_pere SMALLINT UNSIGNED,
  PRIMARY KEY (id_famille)
);
```

```
CREATE TABLE Article (
  id_article     INTEGER NOT NULL AUTO_INCREMENT,
  infos          VARCHAR(100),
  libelle        VARCHAR(255) NOT NULL,
  nom            VARCHAR(50) NOT NULL,
  p_achat        FLOAT,
  p_vente        FLOAT NOT NULL,
  qte_min        INTEGER,
  qte_stock      INTEGER NOT NULL,
  id_famille     SMALLINT UNSIGNED NOT NULL,
  id_fournisseur SMALLINT,
  PRIMARY KEY (id_article)
);

CREATE TABLE Ligne_Facture (
  id_facture  INTEGER NOT NULL,
  id_article  INTEGER NOT NULL,
  qte_article INTEGER NOT NULL,
  subtotal    FLOAT NOT NULL,
  PRIMARY KEY (id_facture, id_article)
);

CREATE TABLE Facture (
  Id_facture    INTEGER NOT NULL AUTO_INCREMENT,
  nbr_ligne     INTEGER NOT NULL,
  montant_total FLOAT NOT NULL,
  date_valid    DATE NOT NULL,
  id_user       SMALLINT UNSIGNED NOT NULL,
  PRIMARY KEY (Id_facture)
);

ALTER TABLE Famille_Article ADD INDEX extrait
(id_famille_pere), ADD CONSTRAINT extrait FOREIGN
KEY (id_famille_pere) REFERENCES Famille_Article
(id_famille);
```

```
ALTER TABLE Article ADD INDEX nomFamille
(id_famille), ADD CONSTRAINT nomFamille FOREIGN KEY
(id_famille) REFERENCES Famille_Article
(id_famille);

ALTER TABLE Article ADD INDEX fournit
(id_fournisseur), ADD CONSTRAINT fournit FOREIGN KEY
(id_fournisseur) REFERENCES Fournisseur
(id_fournisseur);

ALTER TABLE Ligne_Facture ADD INDEX appartient
(id_facture), ADD CONSTRAINT appartient FOREIGN KEY
(id_facture) REFERENCES Facture (Id_facture);

ALTER TABLE Ligne_Facture ADD INDEX
appartientArticle (id_article), ADD CONSTRAINT
appartientArticle FOREIGN KEY (id_article)
REFERENCES Article (id_article);

ALTER TABLE Facture ADD INDEX Valide (id_user), ADD
CONSTRAINT Valide FOREIGN KEY (id_user) REFERENCES
`User` (id);
```

L'exécution des scripts se fait directement via l'interpréteur de commande SQL de Netbeans (service Database).

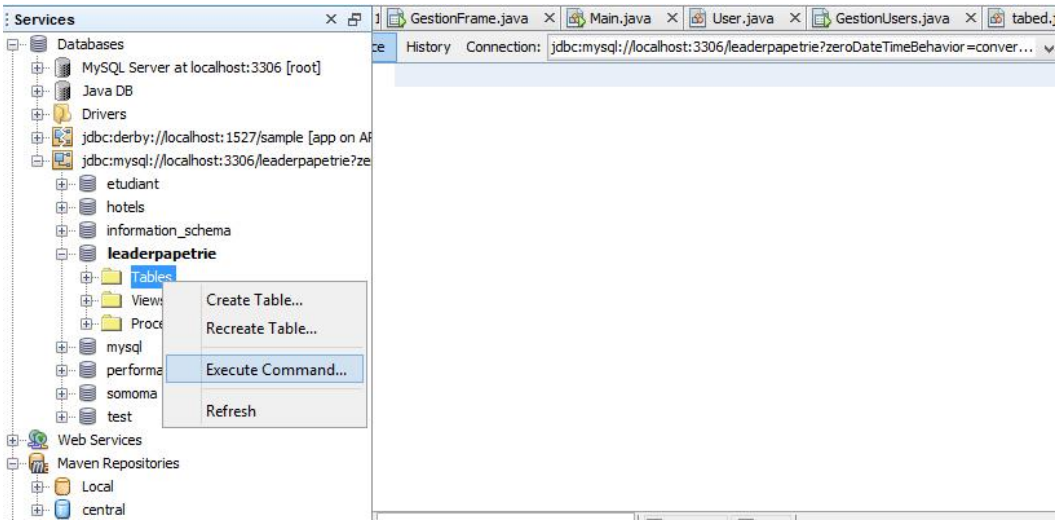

FIGURE 7 - EXECUTION DES SCRIPTS

3- CONFIGURATION HIBERNATE

1-4 CONFIGURATION PRINCIPALE

On commence par ajouter le Framework à notre projet. Sous l'onglet librairie ajouter librairie après un clic droit donnera la figure suivante ou on peut choisir le Framework Hibernate.

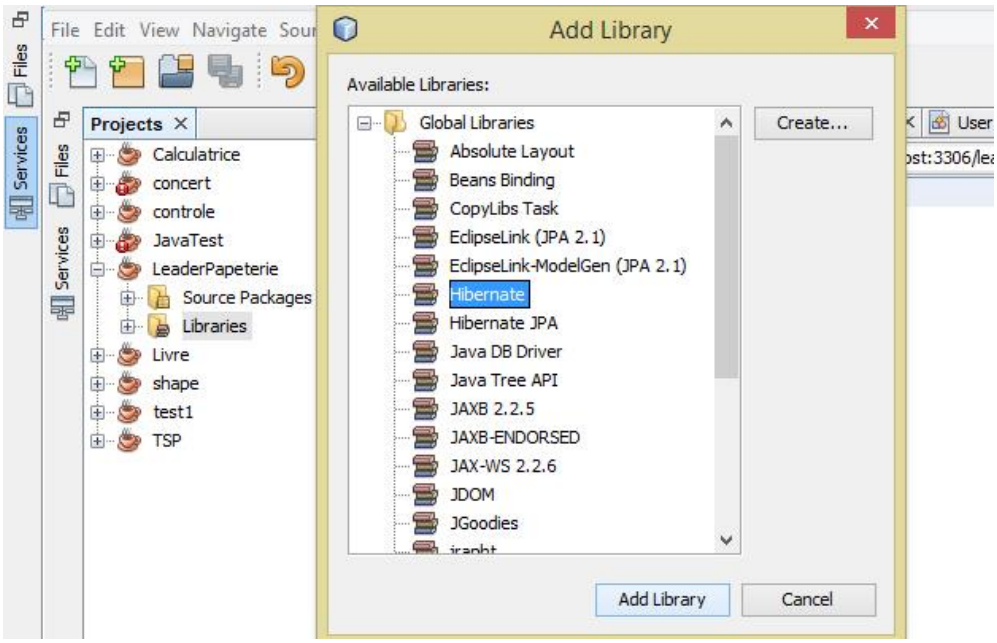

FIGURE 8 - AJOUTER HIBERNATE

Lorsqu'on crée un nouveau projet qui utilise le framework Hibernate, l'IDE crée automatiquement le fichier de configuration hibernate.cfg.xml à la racine du chemin de classe de contexte de l'application (dans la fenêtre Fichiers, src / java).

Dans notre cas il faut créer ce fichier manuellement, et le configurer, voir figures suivantes.

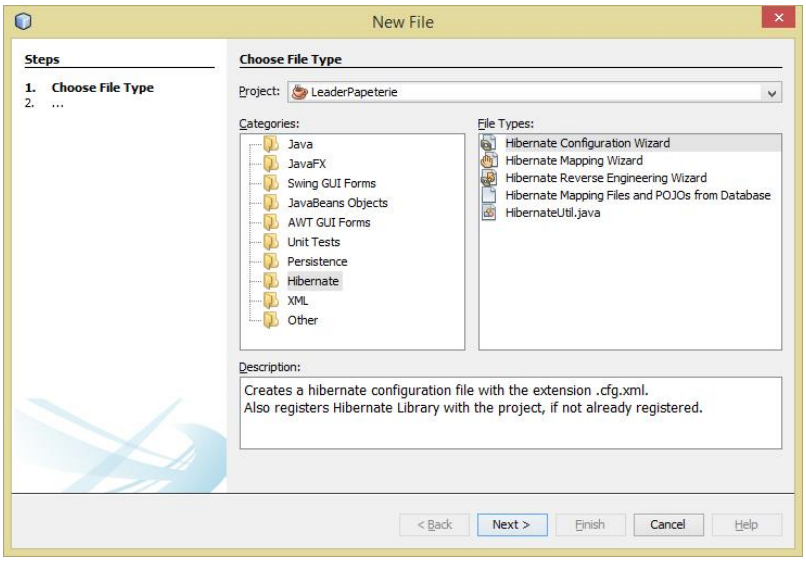

FIGURE 9 - CREATION DU FICHIER DE CONFIGURATION

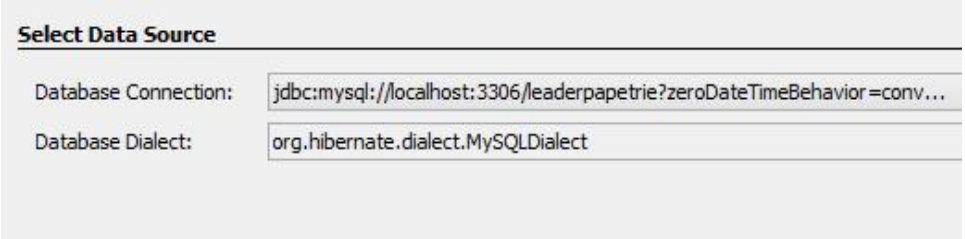

FIGURE 10 - CHOIX DE LA CONNECTION ET DU LANGAGE

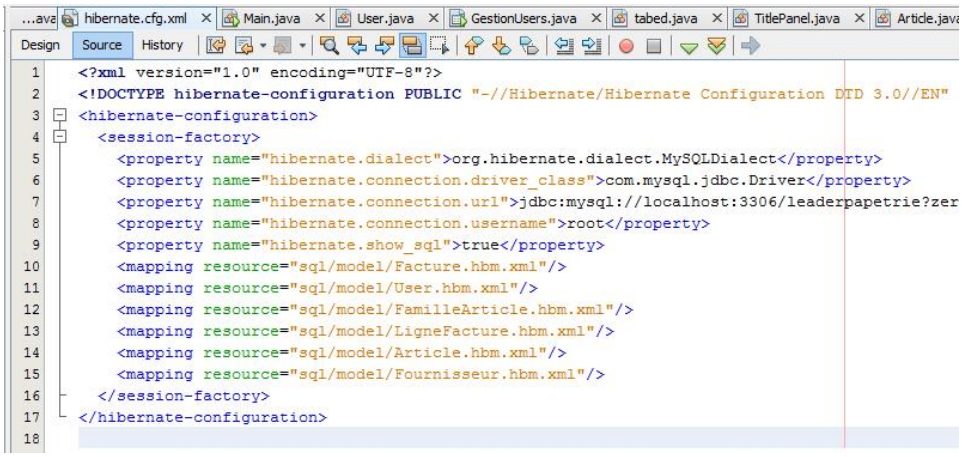

FIGURE 11 - NOTRE FICHIER DE CONFIGURATION.XML

Reste maintenant à créer le fichier de Reverse Engineering et générer les classes.

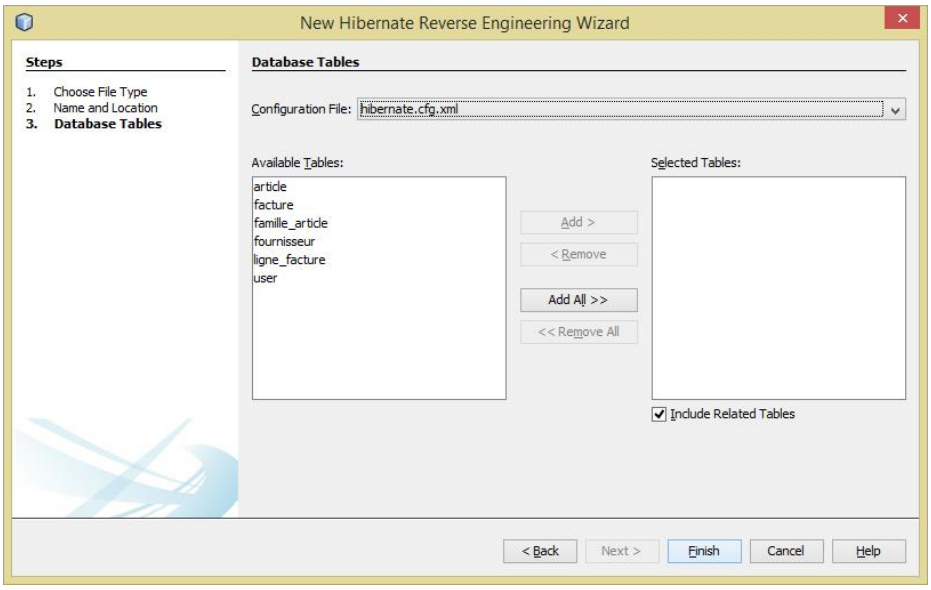

FIGURE 12 - CREATION DU FICHIER REVERSE ENGINEERING

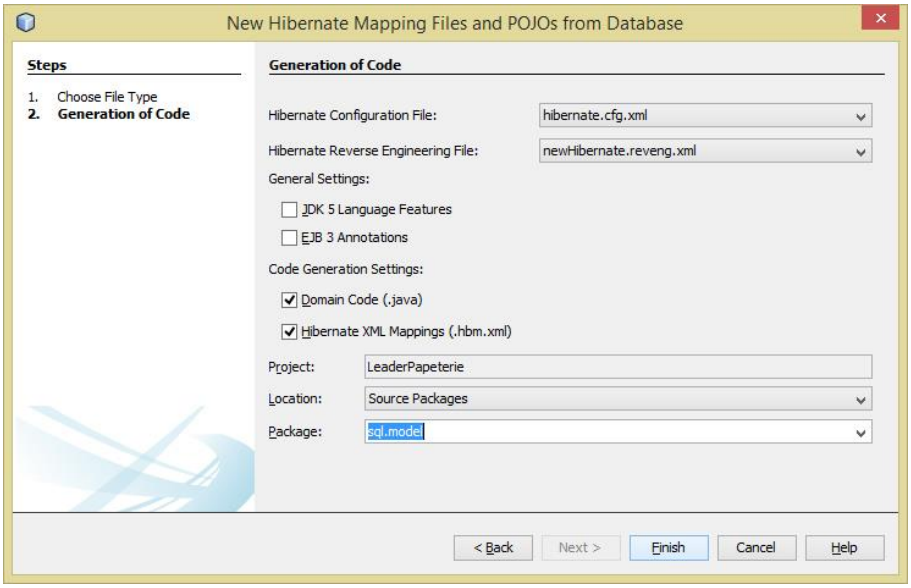

FIGURE 13 - GENERATION DES MAPS ET CLASSES

Nos classes sont prêtes à être utilisées.

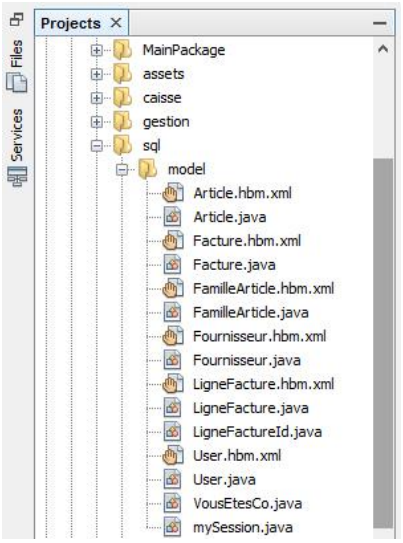

FIGURE 14 - LISTE DES CLASSES

```xml
<hibernate-mapping>
    <class name="sql.model.Fournisseur" table="fournisseur" catalog="leaderpape
        <id name="idFournisseur" type="java.lang.Short">
            <column name="id_fournisseur" />
            <generator class="identity" />
        </id>
        <property name="nomFournisseur" type="string">
            <column name="nom_fournisseur" length="50" not-null="true" />
        </property>
        <property name="tel" type="string">
            <column name="tel" length="40" />
        </property>
        <property name="adr" type="string">
            <column name="adr" />
        </property>
        <property name="mail" type="string">
            <column name="mail" length="100" />
        </property>
        <set name="articles" inverse="true">
            <key>
                <column name="id_fournisseur" />
            </key>
            <one-to-many class="sql.model.Article" />
        </set>
    </class>
</hibernate-mapping>
```

FIGURE 15 - FICHIER MAPPAGE

```java
public class Fournisseur implements java.io.Serializable {

    private Short idFournisseur;
    private String nomFournisseur;
    private String tel;
    private String adr;
    private String mail;
    private Set articles = new HashSet(0);

    public Fournisseur() {
    }

    public Fournisseur(String nomFournisseur) {
        this.nomFournisseur = nomFournisseur;
    }

    public Fournisseur(String nomFournisseur, String tel, String adr, String ma
        this.nomFournisseur = nomFournisseur;
        this.tel = tel;
        this.adr = adr;
        this.mail = mail;
        this.articles = articles;
    }

    public Short getIdFournisseur() {
        return this.idFournisseur;
    }
```

FIGURE 16 - CLASSE JAVA

4- INTERFACES GRAPHIQUE

NetBeans propose un environnement graphique exploitant Swing, désigné comme Java Swing GUI Builder (ancien projet Matisse) 18. Cet environnement comporte des palettes d'outils de composition d'interfaces (composantes Swing et AWT et composantes spécifiques développées pour les projets). Il est utilisé au travers d'une interface graphique simple (choisir et positionner) qui permet de dessiner des interfaces pour les utilisateurs et de les visualiser à la volée. La construction d'interface graphique est conforme à JSR 296 (Swing Application Framework), et JSR 295 (technologie Beans Binding).

Même avec le GUI Builder ont a trouvé quelque difficulté d'intégrité, ce qui nous à ramener à programmer quelque UI en ligne de code, ci-dessous quelques exemple d'interfaces.

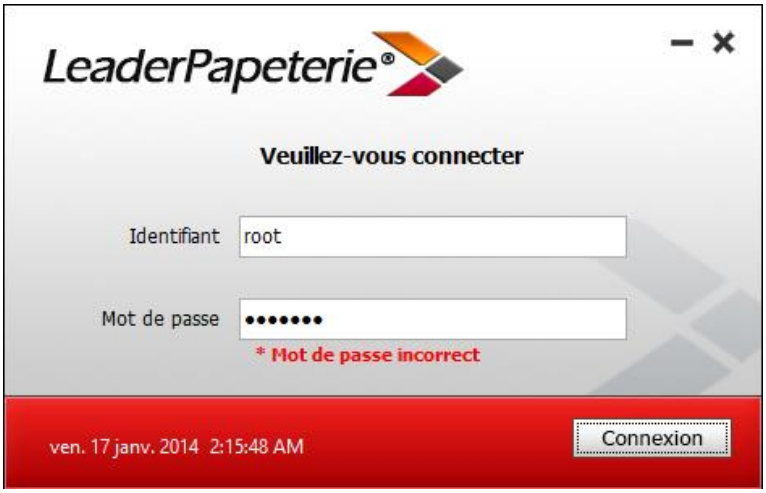

FIGURE 17 - FENETRE D'AUTHENTIFICATION

FIGURE 18 - FENETRE PRINCIPALE

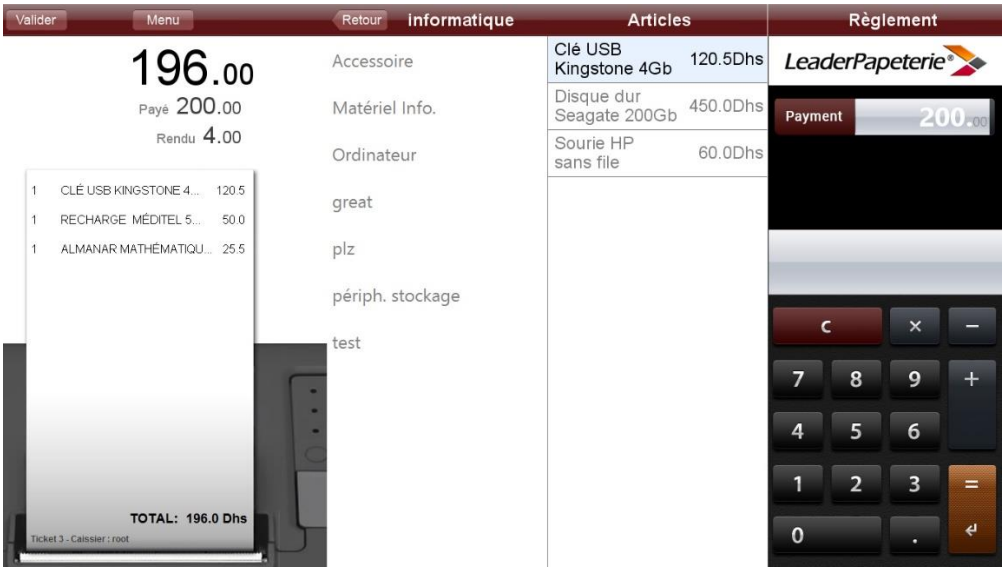

FIGURE 19 - CAISSE

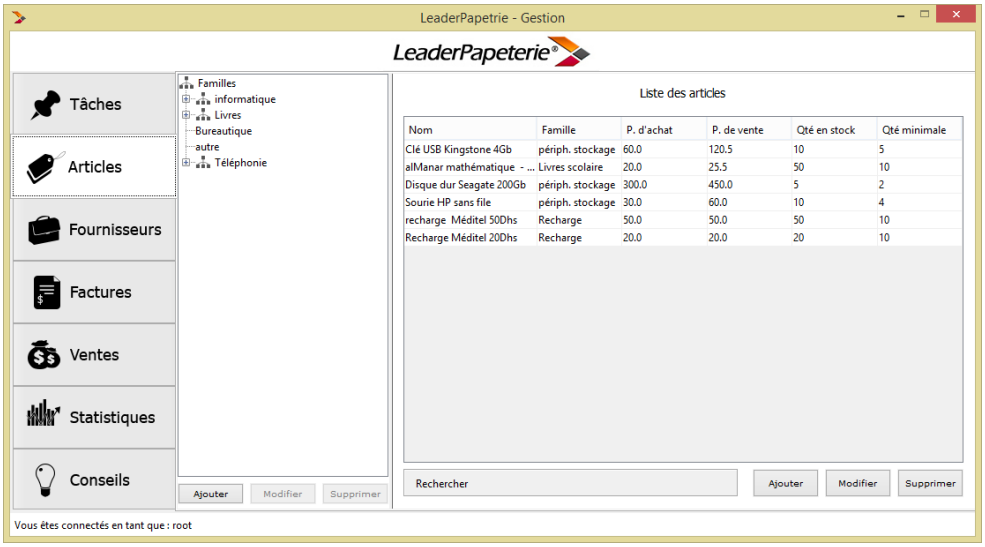

FIGURE 20 - FENETRE GESTION

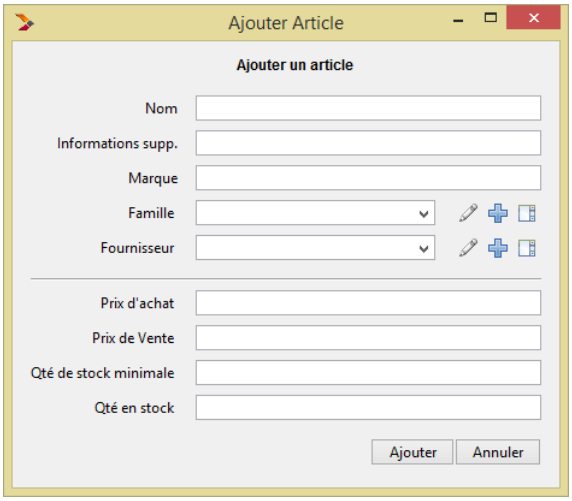

FIGURE 21 - AJOUTER ARTICLE

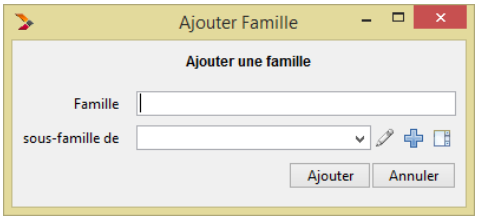

FIGURE 22 - AJOUTER UNE FAMILLE

CONCLUSION

Notre projet est toujours en cours de finalisation, le module le plus stable et immune aux bugs est la caisse, nous travaillons toujours sur les autres modules, notre motivation ne s'arrête pas en cette petite tentative. Notre prochain but après la finalisation de l'application et de pourvoir la commercialiser avec plus d'option et de modules (lecteur code barre, imprimante des tickets, multi caisses liées au même serveur…).

C'est notre première expérience de développement d'une application en groupe, c'était un forum d'échange de connaissance et d'expérience et surtout de fun.

Ce projet nous a permis d'embarquer les différents concepts fréquentés pendant les cours de programmation orienté objet JAVA et de les mettre en application tout au long de son croissance.

PROJET DE SEMESTRE

PROGRAMMATION ORIENTEE OBJET - JAVA

LEADERPAPETRIE

Etant donné l'importance que nous accordons à l'achat des fournitures scolaire, une facilité de gestion des librairies pour une meilleure exploitation s'impose d'où la nécessité de leur proposer une solution plus optimale, moins couteuse, plus performante, et plus prémunie de tout ce qui est erreur. Une solution intégrante, utilisant des outils pour permettre la gestion de la caisse au complet.

Les libraires ont besoin d'un outil permettant un suivi rigoureux du stock, de la caisse des produits…, s'adapter aux exigences du marché et de la clientèle de manière efficace. C'est là qu'intervient notre projet dont le but est de créer une application Java Desktop de gestion de point de vente d'une librairie. Notre logiciel intitulé ***LeaderPapeterie*** saura répondre à tous les besoins listé ci-dessus, puisqu'il va assurer :

- La gestion de caisse (impression facture, retenu du montant, remise du change…).
- La gestion des articles, du stock, des fournisseurs…

ENSA
MARRAKECH

Elle est un acteur public qui exerce le rôle de leadership dans les filières éducatives d'ingénierie à haute valeur ajoutée, assurant une formation de qualité et de développement de compétences, construit sur un engagement partagé entre les élèves ingénieur, les professeurs et le corps administratif et technique.

GI2 - 2013/2014