

République Algérienne Démocratique et Populaire

Université Abou Bakr Belkaid–Tlemcen

Faculté des Sciences Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Licence en Informatique

Thème

**Réalisation d'une application Client/serveur
avec Swing et Hibernate**

Réalisé par :

- BELBACHIR Imène

- BOUAYAD DEBBAGH Saliha Nesrine

Encadré par :

Mr BENMAMMAR Badr

Présenté le 27 Juin 2013 devant la commission d'examen composée de :

- Mr BENMAMMAR Badr (Examineur)
- Mr MERZOUG (Examineur)
- Mr BELABED Amine (Examineur)

Année universitaire: 2012-2013

Résumé :

Toute pharmacie doit être bien gérée pour faciliter la vente de n'importe quel produit, pour cela nous avons présenté une application client/serveur pour la gestion de pharmacie de notre faculté en se basant sur swing et Hibernate .

Dans le premier chapitre nous avons défini tous les outils utilisés dans ce PFE, en commençant par le langage de programmation « JAVA », ensuite l'EDI « NetBeans » et après le système de gestion de bases de données relationnelles « MySQL », ensuite nous avons donné une description générale des « frameworks » en se basant sur « Hibernate », et en fin nous avons présenté les différents types de « l'architecture client/serveur ».

Le deuxième chapitre consiste à présenter toutes les étapes de l'application réalisée dans le cadre de ce PFE.

Abstract:

Any given pharmacy should be well maintained by performed gestion, to ease drugs sales for that we have presented an application client / server in order to run the pharmacy of our faculty, based on "swing" and "Hibernate".

In this first chapter we have defined all the tools used in the Project, by focusing on the programming language "Java", then "NetBeans" after the assumed data base system "MySQL", moreover we have delineated a general description of " frameworks" based on " Hibernate ", at the end we have presented the different types of "client / server" architecture.

Indeed, the second chapter consists presenting all the improved application steps in this finally study work.

ملخص

ينبغي على كل صيدلية ان تدار بشكل جيد لتسهيل بيع أي منتج، لهذا الغرض قمنا بتقديم تطبيق زبون / خادم لإدارة صيدلية الكلية اعتمادا على "سوينج" و "هيبيرنت".

في الفصل الأول قمنا بتعريف كل الأدوات المستخدمة في مشروعنا ، بدءا من لغة البرمجة "جاوا"، ثم التبادل الإلكتروني للبيانات "نتبينس" وبعد ذلك نظام إدارة قواعد البيانات العلائقية "ماي اس كوال"، ثم القينا وصفا لل "فرايم وورك"، "هيبيرنت" و في النهاية قمنا بتقديم مختلف انواع تطبيق زبون / خادم.

اما الفصل الثاني فيتضمن جميع المراحل اللازمة للتطبيق و تشغيله و في النهاية اختتمنا بإظهار أنواع التنفيذات .

Remerciements

Remerciements

Nous tenons tout d'abord à remercier Dieu 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 Mr BENMAMMAR pour son précieux conseil et son aide durant toute la période du travail et de son orientation, sa confiance, sa patience qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené au bon port.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail.. Et de l'enrichir par leurs propositions. Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

DEDICACE

Ce travail, et bien au-delà, je le dois à mes très chers parents qui m'ont fourni au quotidien un soutien et une confiance sans faille et de ce fait, je ne saurais exprimer ma gratitude seulement par des mots. Que dieu vous protège et vous garde pour no us.

A ma chère sœur, mes trois chers frères, mes adorables amies, mon binôme qui m'a supporté durant ces trois dernières années.et chez qui j'ai trouvé l'entente dont j'avais besoin.

A tous mes amis avec lesquels j'ai partagé mes moments de joie et de bonheur.
Que toute personne m'ayant aidé de près ou de loin, trouve ici l'expression de ma reconnaissance.

BELBACHIR Imène

DEDICACE

A la mémoire de mon père qui a souhaité vivre afin de nous voir réussir dans la vie.

A celle qui m'a transmis la vie, l'amour, le courage, à toi chère maman je te dois toutes mes joies, mon amour et ma reconnaissance.

A ma petite sœur, mon frère, mes adorables amies, mon binôme qui m'a supporté durant ces trois dernières années. et chez qui j'ai trouvé l'entente dont j'avais besoin.

A tous mes amis avec lesquels j'ai partagé mes moments de joie et de bonheur.
Que toute personne m'ayant aidé de près ou de loin, trouve ici l'expression de ma reconnaissance.

BOUAYAD DEBBAGH Nesrine

Table des matières

Sommaire

I. Introduction générale	1
II. Architecture et outils utilisés	4
II.1 Introduction :.....	4
II.2 JAVA.....	4
II.2.1 Définition :	4
II.2.2 Définition d'un plug-in Java :	4
II.2.3 JVM : machine virtuelle Java :	5
II.2.4 Les caractéristiques de Java :	5
II.2.5 Des outils.....	5
II.2.8 Programmation :.....	5
II.3 NetBeans :	6
II.3 .1 Historique :.....	6
II.3.2 Définition :	6
II.3.3 Environnement de base :.....	6
II.3.4 Bases de données :	6
II.3.5 Débogage et optimisation :	7
II.4 EasyPHP :	7
II.4.1 MySQL :.....	7
II.5 Interface graphique:.....	7
II.5.1 Introduction:	7
II.5.2 AWT :.....	8
II.5.3 SWING :	8
II.6 Framework:	8
II.7 Hibernate:.....	9
II.7.1. Les besoins :.....	9
II.7.2. Comparaison :	10
II.7.3. Architecture :	10
II.7.4 Génération de la base de données :	10
II.8 Architecture Client-serveur :	11
II.8.1 Présentation de l'architecture d'un système client/serveur	11
II.8.2 Avantages de l'architecture client/serveur :	11

Table des matières

II.8.3 Inconvénients du modèle client/serveur :.....	11
II.8.4 Fonctionnement d'un système client/serveur :	11
II.8.5 Les différentes architectures du client/serveur :	12
II.9 Conclusion :	14
III. Présentation de l'application	16
III.1 Introduction :.....	16
III.2 Création de la base de données :	16
III.3 Accéder à notre base de données à partir de NetBeans :.....	16
III.4 Création du projet :	18
III.4.1 Créer un projet Swing :	18
III.4.2 L'ajout du framework « Hibernate » dans la librairie	19
III.4.3 Arborescence du projet	20
III.4.4 Créer le fichier de configuration hibernate.cfg.xml :.....	20
III.4.5 Modifier hibernate.cfg.xml:	23
III.4.6 Créer le « Helper file » : fichier HibernateUtil .java :.....	23
III.4.7 Générer les fichiers de mapping et de classes java:	25
III.4.8 Créer le fichier « Reverse Engineering » :.....	25
III.4.9 Créer «Hibernate Mapping Files and POJOs From a Database :	28
III.5 Créer une JFrame pour faire des recherches dans la table des produits d'une pharmacie :.....	32
III.5.1 Créer la JFrame :	32
III.5.2 Ajouter des éléments a la JFrame :.....	34
III.5.3 Changeant le nom des variables :.....	35
III.5.4 Ajouter les déclarations suivantes au code source :	35
III.5.5 Ajouter les méthodes suivantes, qui traitent le caractères saisi à partir de l'IHM :	36
III.5.6 Ajouter la méthodeexecuteHQLQuery() :.....	36
III.5.7 Ajouter les imports nécessaires:	36
III.5.8 Basculer vers Design et double clic sur le bouton Recherche :.....	36
III.5.9 Ajouter la méthode displayresult :	37
III.5.10 Ajouter les imports nécessaires :.....	38
III.6 Créer le JDialog pour faire la recherche par forme et la recherche par péremption : .	38

Table des matières

III.6.1 Créer et ajouter des éléments a la JDialog :	38
III.6.2 Le code source :	40
III.6.3 Astuce de plus ! :	40
III.7 Exécution :	41
III.7.1 Premier mode d'exécution :	41
III.7.2 deuxième mode d'exécution :	42
III.7.3 troisième mode d'exécution	43
III.7.4 Quatrième mode d'exécution :	43
III.7.5 cinquième mode d'exécution	44
III.7.6 Sixième mode d'exécution :	45
III.8 Conclusion :	45

Liste des figures

Figure 1 : Architecture d’Hibernante [18]	10
Figure 2: Le fonctionnement d’un système client/serveur [19]	12
Figure 3 : Architecture a deux niveaux [20]	13
Figure 4: La table pharmacie	16
Figure 5 : Connecter a la base de données	17
Figure 6: Connecter à la base de données	17
Figure 7 : Affichage de la table sous NetBeans	18
Figure 8 : Résultat de l’affichage de la table.....	18
Figure 9: Créer le projet.....	19
Figure 10 : Ajout la librairie Hibernate.....	19
Figure 11 : Arborescence du projet.....	20
Figure 12 : Créer le fichier de configuration(1/3).....	20
Figure 13 : Créer le fichier de configuration (2/3).....	21
Figure 14 : Créer le fichier de configuration (3/3).....	21
Figure 15 : Contenu design de fichier de configuration.....	22
Figure 16 : Contenu XML de fichier de configuration	22
Figure 17 : Créer le Helper File (1/2)).....	23
Figure 18 : Créer le Helper File (2/2)	24
Figure 19 : Créer le fichier Reverse Engineering(1/2).....	25
Figure 20 : Créer le fichier Reverse Engineering(2/2).....	26
Figure 21 : Sélection de la table pharmacie	26
Figure 22 : Ajouter la table pharmacie.....	27
Figure 23 : Hibernate Mapping Files and POJOs from a Database(1/2)	28
Figure 24 : Hibernate Mapping Files and POJOs from a Database(2/2)	28
Figure 25 : arborescence du projet.....	29
Figure 26 : Test avec des requêtes HQL	32
Figure 27 : Créer la JFrame (1/2).....	32
Figure 28 : Créer la JFrame (2/2).....	33
Figure 29 : Zone de la JFrame	33
Figure 30 : Interface principale.....	34
Figure 31 : Créer la JDialog(1/2)	37

liste des figures

Figure 32 : Créer la JDialog(2/2)	38
Figure 33 : Zone de JDialog.....	38
Figure 34 : Interface recherche par forme	39
Figure 35 : Interface recherche par date de péremption.....	39
Figure 36 : Interface de mot de passe.....	40
Figure 37 : Premier mode d'exécution.....	41
Figure 38 : Deuxième mode d'exécution	41
Figure 39 : Troisième mode d'exécution	42
Figure 40 : Quatrième mode d'exécution.....	43
Figure 41 : Cinquième mode d'exécution.....	44
Figure 42 : Sixième mode d'exécution	45

INTRODUCTION GENERALE

I. Introduction générale

L'arrivée du contexte de la gestion via l'outil informatique se concrétisa par l'arrivée de l'orienté objet. Voir les résolutions des problèmes avec des langages de programmation comme JAVA.

L'outil informatique a connu beaucoup d'évolutions technologiques dans tous les domaines, la réalisation des applications professionnelles demeure comme étant une activité difficile, et qui répond d'une manière lente aux besoins, contrairement à plusieurs langages différents peuvent être utilisés pour élaborer différents logiciels dont le but est commun .

Notre travail consiste à représenter globalement la conception d'un gestionnaire et qui s'intitule : réalisation d'une application client/serveur (gestion d'une pharmacie) avec swing et hibernate .

Ce travail est divisé en deux parties :

- ❖ **CHAPITRE 1** : qui est une partie théorique, nous allons présenter d'une manière générale les différents outils que nous allons utiliser : une introduction au langage JAVA , suivi par une présentation de Netbeans qui est un environnement de développement intégré (EDI), ensuite nous allons présenter MySQL qui est un serveur de base de données relationnelles , suivi d'une description du framework « Hibernate », enfin nous allons présenter l'architecture client/serveur utilisé dans le cadre de ce PFE.
- ❖ **CHAPITRE 2** : qui est une partie pratique contenant les étapes nécessaires pour la réalisation de notre application.

CHAPITRE 1 :
ARCHITECTURE ET
OUTILS UTILISES

II. Architecture et outils utilisés :

II.1 Introduction :

Dans cette partie , nous allons faire des petites présentation sur tous les outils que nous allons utiliser , en commençant par le langage JAVA ,suivi par la présentation de NetBeans qui est un environnement de développements (EDI), ensuite nous allons présenter MySQL qui est un serveur de base de données relationnelles ,ensuite nous allons évoquer quelques composants de la bibliothèque SWING , ensuite nous passons à une description des frameworks qui appartient au java , nous allons se baser en particulier sur le framework « Hibernate » qui est utilisé dans ce projet , et enfin nous allons faire un petit passage sur l'architecture Client/serveur.

II.2 JAVA

II.2.1 Définition :

Java est un langage de programmation orienté objets, qui permettent d'exécuter des programmes au travers d'une machine virtuelle. Il est fourni avec un ensemble d'outils (le JDK Java Development Kit) et un ensemble de packages : ensemble de classes. Ces différentes classes de base couvrent beaucoup de domaine (entrées/sorties, interface graphique, réseau, etc.) Cette richesse en "bibliothèques standards" explique sûrement en partie le succès de Java. Le langage lui-même se trouve dans le package java.lang , Java est donc :

- Un langage de programmation objets.
- Une architecture de machine virtuelle.
- Un ensemble d'outils.
- Un ensemble de bibliothèques (packages) de base.

Java n'a rien de commun avec HTML. Ce n'est pas un langage de script (type PERL ou TCL). Il ne sert pas seulement à faire des applets pour le WEB, mais ce la a contribué à son succès.[1]

II.2.2 Définition d'un plug-in Java :

Le plug-in Java est un composant de l'environnement d'exécution Java (JRE). JRE permet aux applets écrits dans le langage de programmation Java de s'exécuter dans différents navigateurs. Le plug-in Java n'est pas un programme autonome et ne peut pas être installé séparément. [2]

II.2.3 JVM : machine virtuelle Java :

La machine virtuelle Java n'est qu'un aspect du logiciel Java, lié à l'interaction avec le Web. La machine virtuelle Java est incluse dans le téléchargement du logiciel Java et permet l'exécution des applications Java.[3]

II.2.4 Les caractéristiques de Java :

- ❖ Un langage orienté objets.
- ❖ Portabilité.
- ❖ Robustesse.
- ❖ Sécurité.
- ❖ Multitâche.
- ❖ Client-serveur.
- ❖ Application indépendante et applet.
- ❖ Les performances.[4]

II.2.5 Des outils

❖ Les outils du JDK : Java Development Kit : Le JDK comprend entre autres

- javac : compilateur de sources java
- java : interpréteur de byte code
- appletviewer : interpréteur d'applet
- javadoc : générateur de documentation (HTML, MIF), très intéressant, permet de générer une documentation standardisée des classes.

Java n'est pas réservée à la programmation d'applets pour le WEB. Il semble constituer une véritable alternative pour le développement d'applications importantes même si il manque encore de références en la matière. [5]

II.2.8 Programmation :

La programmation peut se faire pour des exemples simples avec le compilateur javac, mais pour avoir plus de confort il est préférable s'utiliser un environnement de développement intégré ou IDE, comme Eclipse ou NetBeans . Dans notre projet nous avons utilisé NetBeans.[6]

II.3 NetBeans :

II.3.1 Historique :

En 1997, NetBeans naît de Xelfi, un projet d'étudiants dirigé par la Faculté de mathématiques et de physique de l'Université Charles de Prague. Plus tard, une société se forme autour du projet et édite des versions commerciales de l'EDI NetBeans, jusqu'à ce qu'il soit acheté par Sun en 1999. Sun place le projet sous double licence CDDL et GPL v2 en juin de l'année suivante.[7]

II.3.2 Définition :

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).

NetBeans constitue par ailleurs une plate-forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plate-forme. L'IDE Netbeans s'enrichit à l'aide de plugins.[8]

II.3.3 Environnement de base :

L'environnement de base comprend les fonctions générales suivantes

- configuration et gestion de l'interface graphique des utilisateurs,
- support de différents langages de programmation,
- fonctions d'import/export depuis et vers d'autres IDE, tels qu'Eclipse ou JBuilder,
- accès et gestion de bases de données, serveurs Web, ressources partagées.[9]

II.3.4 Bases de données :

NetBeans comprend un explorateur de bases de données qui supporte toutes les bases relationnelles pour lesquelles un connecteur JDBC existe (selon les versions des gestionnaires de bases de données) : JavaDB (Derby) MySQL, PostgreSQL, Oracle, Microsoft SQL, PointBase, jTDS, IBM Redistributable DB2, ...[10]

II.3.5 Débogage et optimisation :

Netbeans comprend un profileur Java (analyse des performances CPU, de la génération de charge, analyse de l'utilisation mémoire, ...). Il intègre par ailleurs des outils de débogage Java. Il peut aussi interagir avec des débogueurs PHP (Zend, Xdebug).[11]

II.4 EasyPHP :

C'est un environnement comprenant un serveur Web (Apache), un serveur SQL (MySQL), un interpréteur de script PHP et un administrateur de base SQL (phpMyAdmin). Cet environnement permet de faire fonctionner en local un site Internet développé en PHP sans à avoir à se connecter à un serveur externe. Cela peut être utile dans le cadre du développement d'un site pour tester différentes versions ou pour rendre accessible un site Internet sur un réseau fermé, comme un Intranet. EasyPHP est gratuit et s'installe comme une application même si ce n'est pas une application. [12]

II.4.1 MySQL :

MySQL est une base de données relationnelle libre qui a vu le jour en 1995 et très employée sur le Web, souvent en association avec PHP (langage) et Apache (serveur web). MySQL fonctionne indifféremment sur tous les systèmes d'exploitation (Windows, Linux, Mac OS notamment).

Le principe d'une base de données relationnelle est d'enregistrer les informations dans des tables, qui représentent des regroupements de données par sujets (table des clients, table des fournisseurs, table des produits, par exemple). Les tables sont reliées entre elles par des relations.

Le langage SQL (acronyme de Structured Query Language) est un langage universellement reconnu par MySQL et les autres bases de données et permettant d'interroger et de modifier le contenu d'une base de données. Les autres bases de données utilisées en informatique sont essentiellement Microsoft SQL Server et Oracle.[13]

II.5 Interface graphique:

II.5.1 Introduction:

Il existe deux principaux types de composants susceptibles d'intervenir dans une interface graphique : les conteneurs qui sont destinés à contenir d'autres composants, comme par exemple les fenêtres, les composants atomiques qui sont des composants qui ne

peuvent pas en contenir d'autres, comme par exemple les boutons. Les deux bibliothèques les plus utilisés sous JAVA sont AWT et SWING. [14]

II.5.2 AWT :

Abstract Window Toolkit (AWT) est une bibliothèque graphique pour Java, faisant partie de JFC (Java Foundation Classes). Cette bibliothèque a été introduite dès les premières versions de Java ; depuis Java 2, la bibliothèque de gestion de fenêtre officielle est Swing. Toutefois, AWT sert encore de fondement à Swing, dans la mesure où de nombreuses classes Swing héritent de classes AWT.

AWT emploie les composants natifs de la plate-forme, alors que Swing utilise des composants en pur Java.[15]

II.5.3 SWING :

Swing est une bibliothèque graphique pour le langage de programmation Java, faisant partie du package Java Foundation Classes (JFC), inclus dans J2SE. Swing constitue l'une des principales évolutions apportées par Java 2 par rapport aux versions antérieures.

Swing offre la possibilité de créer des interfaces graphiques identiques quel que soit le système d'exploitation sous-jacent, au prix de performances moindres qu'en utilisant Abstract Window Toolkit (AWT). Il utilise le principe Modèle-Vue-Contrôleur (MVC, les composants Swing jouent en fait le rôle du contrôleur au sens du MVC) et dispose de plusieurs choix d'apparence (de vue) pour chacun des composants standards.[16]

II.6 Framework:

Le terme Framework est fréquemment utilisé dans des contextes différents mais il peut être traduit par cadre de développement.

Les Framework se présentent sous diverses formes, qui peuvent inclure tout ou partie des éléments suivants :

- Un ensemble de classes généralement regroupées sous la forme de bibliothèques pour proposer des services plus ou moins sophistiqués.
- Un cadre de conception reposant sur les design patterns pour proposer tout ou partie d'un squelette d'applications.
- Des recommandations sur la mise en œuvre et des exemples d'utilisation.
- Des normes de développement.
- Des outils facilitant la mise en œuvre.

L'objectif d'un Framework est de faciliter la mise en œuvre des fonctionnalités de son domaine d'activité. Il doit permettre au développeur de se concentrer sur les tâches spécifiques à l'application à développer plutôt qu'à des tâches techniques récurrentes telles que :

- L'architecture de base de l'application.
- L'accès aux données.
- L'internationalisation.
- La journalisation des événements (logging).
- La sécurité (authentification et gestion des rôles).
- Le paramétrage de l'application.

La mise en œuvre d'un Framework permet notamment :

- De capitaliser le savoir-faire sans "réinventer la roue".
- D'accroître la productivité des développeurs une fois le Framework pris en main.
- D'homogénéiser les développements des applications en assurant la réutilisation de composants fiables.
- Donc de faciliter la maintenance notamment évolutive des applications.

Cependant, cette mise en œuvre peut se heurter à certaines difficultés :

- le temps de prise en main du Framework par les développeurs peut être plus ou long en fonction de différents facteurs (complexité du Framework, richesse de sa documentation, expérience des développeurs, ...)
- les évolutions du Framework qu'il faut répercuter dans les applications existantes.[17]

II.7 Hibernate:

II.7.1. Les besoins :

Hibernate est un logiciel écrit sous la responsabilité de Gavin King, qui fait entre autre partie de l'équipe de développement de JBOSS.

L'ensemble des données nécessaires au fonctionnement de l'application sont sauvegardées dans une base de données. La manipulation des données peut se faire de différentes manières : Par l'accès directement à la base en écrivant les requêtes SQL adéquates. Utiliser un outil d'ORM (object Relational Mapping) permettant de manipuler facilement les données et d'assurer leur persistance. Il en existe plusieurs. Hibernate, présenté dans ce document, est l'un des plus utilisés.

Pourquoi ajouter une couche entre l'application et la base de données ? L'objectif est de réduire le temps de développement de l'application en éliminant une grande partie du code SQL à écrire pour interagir avec la base de données et en encapsulant le code SQL résiduel. Les développeurs manipulent les classes dont les données doivent être persistantes comme des classes Java normales. Seules une initialisation correcte d'hibernate doit être effectuée, et quelques règles respectées lors de l'écriture et de la manipulation des classes persistantes.

II.7.2. Comparaison :

Il existe d'autres outils d'ORM permettant d'assurer la persistance de données, comme JDO et TopLink. Pour l'outil d'ORM JDO, il semble qu'il y a quelques problèmes de compatibilité avec les EJB V3. Voilà ce qu'il est écrit sur :

http://stessy.developpez.com/j2ee/hibernate/?page=page_3#L3.3.5.2 :

« Pour information, il semble que la norme EJB V3, fasse un virage important sur les Session et Entity Beans qui deviendraient des POJO (suppression des homes interfaces, des fichiers de descriptions de déploiement ...). La compatibilité ascendante [avec Hibernate] serait assurée. Les tests de EJB V3 serait fait à partir de deux logiciels de mapping O/R (Hibernate et TopLink) mais pas JDO !. Gavin King [Responsable d'hibernate] fait partie du groupe d'expert EJB V3... »

II.7.3. Architecture :

Voici une vue (très) haut niveau de l'architecture d'

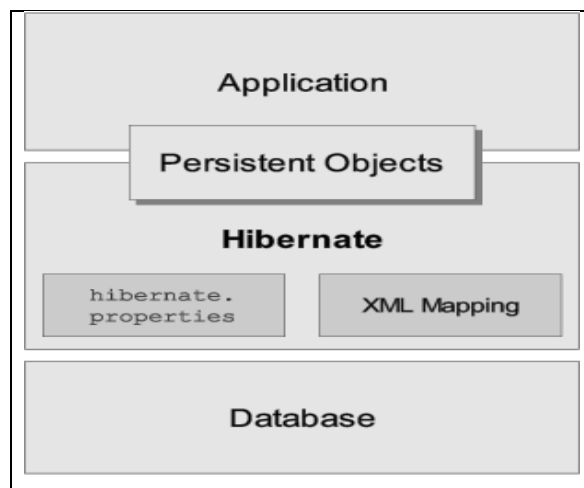


Figure 1 : Architecture d'Hibernate [18]

Ce diagramme montre Hibernate utilisant une base de données et des données de configuration pour fournir un service de persistance (et des objets persistants) à l'application.

L'architecture la plus complète abstrait l'application des APIs JDBC/JTA sous-jacentes et laisse Hibernate s'occuper des détails. [18]

II.7.4 Génération de la base de données :

Il est possible de générer automatiquement le schéma (code SQL) de la base de données à partir des fichiers de mapping des classes.

Commande de génération de la base de données

II.8 Architecture Client-serveur :

II.8.1 Présentation de l'architecture d'un système client/serveur

De nombreuses applications fonctionnent selon un environnement client/serveur, cela signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en termes de capacités d'entrée-sortie, qui leur fournit des services. Ces services sont des programmes fournissant des données telles que l'heure, des fichiers, une connexion, etc.

Les services sont exploités par des programmes, appelés programmes clients, s'exécutant sur les machines clientes. On parle ainsi de client (client FTP, client de messagerie, etc.) lorsque l'on désigne un programme tournant sur une machine cliente, capable de traiter des informations qu'il récupère auprès d'un serveur (dans le cas du client FTP il s'agit de fichiers, tandis que pour le client de messagerie il s'agit de courrier électronique).

II.8.2 Avantages de l'architecture client/serveur :

Le modèle client/serveur est particulièrement recommandé pour des réseaux nécessitant un grand niveau de fiabilité, ses principaux atouts sont :

- Des ressources centralisées : étant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction .
- Une meilleure sécurité : car le nombre de points d'entrée permettant l'accès aux données est moins important.
- Une administration au niveau serveur : les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés.
- Un réseau évolutif : grâce à cette architecture il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modification majeure .

II.8.3 Inconvénients du modèle client/serveur :

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles :

- Un coût élevé dû à la technicité du serveur.
- Un maillon faible : le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui ! Heureusement, le serveur a une grande tolérance aux pannes (notamment grâce au système RAID).

II.8.4 Fonctionnement d'un système client/serveur :

Un système client/serveur fonctionne selon le schéma suivant :

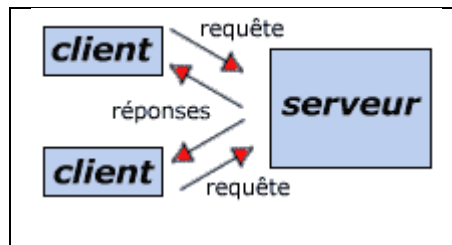


Figure 2 : Le fonctionnement d'un système client/serveur [19]

- Le client émet une requête vers le serveur grâce à son adresse IP et le port, qui désigne un service particulier du serveur
- Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine cliente et son port. [19]

II.8.5 Les différentes architectures du client/serveur :

II.8.5.1 L'architecture 2 tiers :

- Dans une architecture deux tiers, encore appelée client-serveur de première génération ou client-serveur de données, le poste client se contente de déléguer la gestion des données à un service spécialisé. Le cas typique de cette architecture est une application de gestion fonctionnant sous Windows ou Linux et exploitant un SGBD centralisé.
- Ce type d'application permet de tirer parti de la puissance des ordinateurs déployés en réseau pour fournir à l'utilisateur une interface riche, tout en garantissant la cohérence des données, qui restent gérées de façon centralisée.

- La gestion des données est prise en charge par un SGBD centralisé, s'exécutant le plus souvent sur un serveur dédié. Ce dernier est interrogé en utilisant un langage de requête qui, plus souvent, est SQL. Le dialogue entre client et serveur se résume donc à l'envoi de requêtes et au retour des données correspondant aux requêtes.

Malgré tout, l'architecture deux tiers présente de nombreux avantages qui lui permettent de présenter un bilan globalement positif :

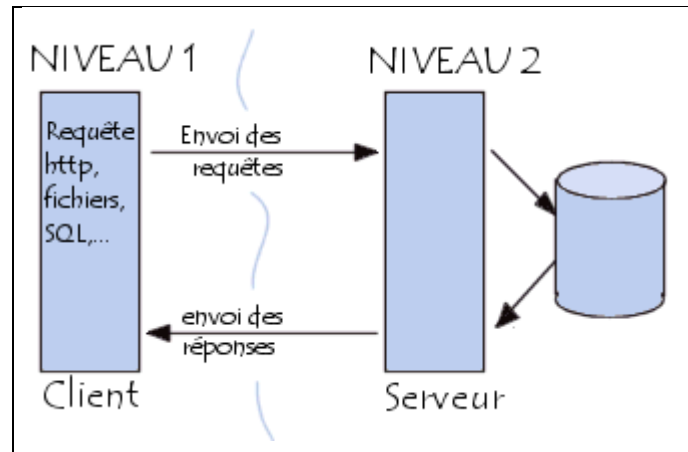


Figure 3 : Architecture a deux niveaux [20]

- Elle permet l'utilisation d'une interface utilisateur riche.
- Elle a permis l'appropriation des applications par l'utilisateur.
- Elle a introduit la notion d'interopérabilité.

II.8.5.2 L'architecture 3 tiers :

Les limites de l'architecture deux tiers proviennent en grande partie de la nature du client utilisé :

- Le frontal est complexe et non standard (même s'il s'agit presque toujours d'un PC sous Windows).
- Le middleware entre client et serveur n'est pas standard (dépend de la plate-forme, du SGBD ...).

La solution résiderait donc dans l'utilisation d'un poste client simple communiquant avec le serveur par le biais d'un protocole standard.

Dans ce but, l'architecture trois tiers applique les principes suivants :

- Les données sont toujours gérées de façon centralisée.
- La présentation est toujours prise en charge par le poste client.
- La logique applicative est prise en charge par un serveur intermédiaire.

Cette architecture trois tiers, également appelée client-serveur de deuxième génération ou client-serveur distribué sépare l'application en 3 niveaux de services distincts, conformes au principe précédent :

- Premier niveau : l'affichage et les traitements locaux (contrôles de saisie, mise en forme de données...) sont pris en charge par le poste client,
- Deuxième niveau : les traitements applicatifs globaux sont pris en charge par le service applicatif,
- Troisième niveau : les services de base de données sont pris en charge par un SGBD.

II.8.5.3 L'architecture n-tiers :

L'architecture n-tiers a été pensée pour pallier aux limitations des architectures trois tiers et concevoir des applications puissantes et simples à maintenir. Ce type d'architecture permet de distribuer plus librement la logique applicative, ce qui facilite la répartition de la charge entre tous les niveaux.

Cette évolution des architectures trois tiers met en œuvre une approche objet pour offrir une plus grande souplesse d'implémentation et faciliter la réutilisation des développements.

Théoriquement, ce type d'architecture supprime tous les inconvénients des architectures précédentes :

- Elle permet l'utilisation d'interfaces utilisateurs riches.
- Elle sépare nettement tous les niveaux de l'application.
- Elle offre de grandes capacités d'extension.
- Elle facilite la gestion des sessions. [21]

II.9 Conclusion :

Dans ce chapitre , nous avons présenté les différents outils utilisés dans notre Projet de fin d'études comme JAVA , NetBeans qui est l'EDI pour le mettre en place, aussi qu'un gestionnaire de bases de données MySQL, suivi de la description d'Hibernate et de différents types d'architecture Client/serveur.

Dans le chapitre suivant, nous présenterons en détail les différentes étapes nécessaires pour la réalisation de notre application « Gestion d'une pharmacie à l'aide d'Hibernate».

CHAPITRE 2 :
PRESENTATION DE
L'APPLICATION

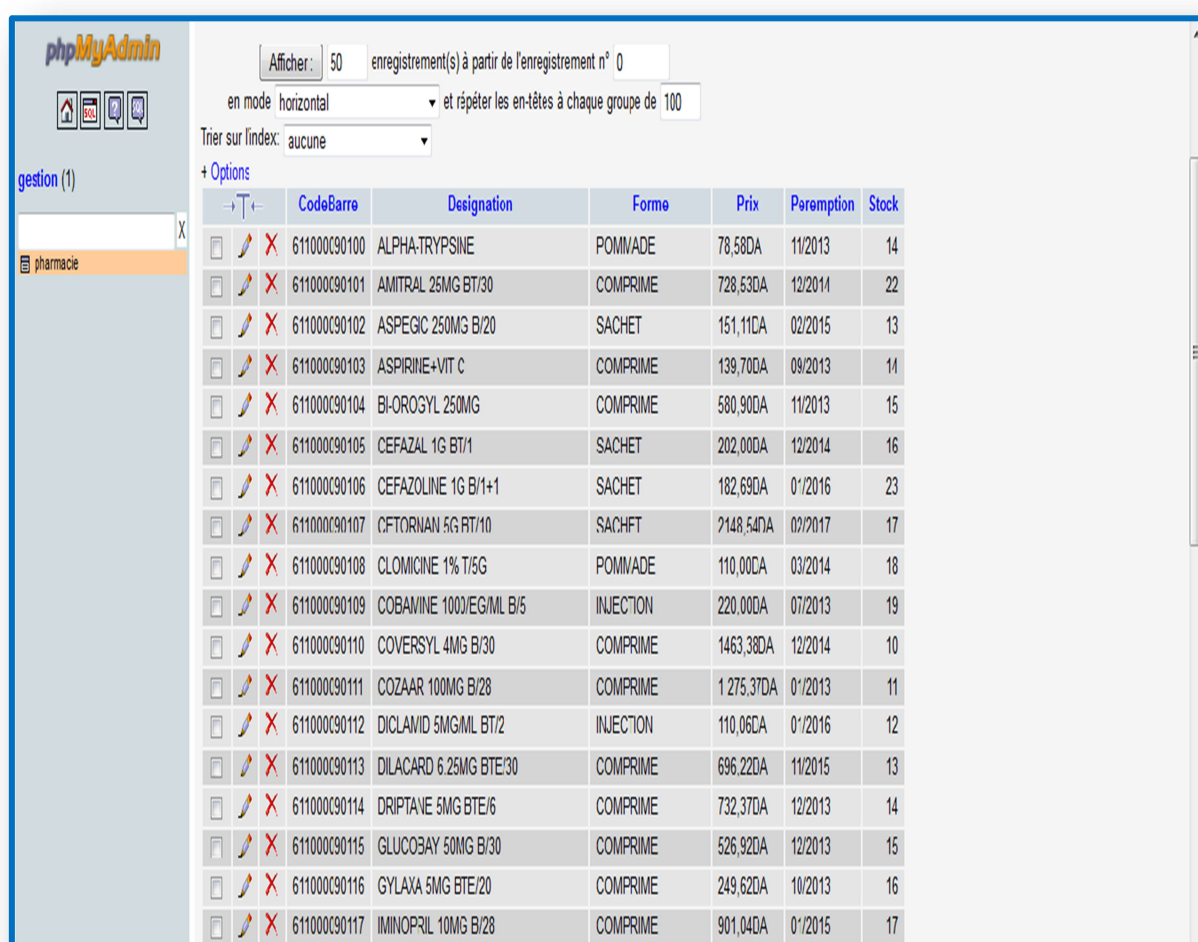
III. Présentation de l'application :

III.1 Introduction :

Dans ce qui suit nous allons présenter les étapes nécessaires pour la réalisation de Notre projet.

III.2 Création de la base de données :

Nous avons créé une table pharmacie à l'intérieur de la base de données gestion, la figure suivante montre le résultat obtenu :



	CodeBarre	Designation	Forme	Prix	Péréemption	Stock
<input type="checkbox"/>	611000C90100	ALPHA-TRYPISINE	POMVADE	78,58DA	11/2013	14
<input type="checkbox"/>	611000C90101	AMITRAL 25MG BT/30	COMPRIME	728,53DA	12/2014	22
<input type="checkbox"/>	611000C90102	ASPEGIC 250MG B/20	SACHET	151,11DA	02/2015	13
<input type="checkbox"/>	611000C90103	ASPIRINE+VIT C	COMPRIME	139,70DA	09/2013	14
<input type="checkbox"/>	611000C90104	BI-OROSYL 250MG	COMPRIME	580,90DA	11/2013	15
<input type="checkbox"/>	611000C90105	CEFAZAL 1G BT/1	SACHET	202,00DA	12/2014	16
<input type="checkbox"/>	611000C90106	CEFAZOLINE 1G B/1+1	SACHET	182,69DA	01/2016	23
<input type="checkbox"/>	611000C90107	CFTORNAV 5G RT/10	SACHFT	2148,54DA	02/2017	17
<input type="checkbox"/>	611000C90108	CLOMOCINE 1% T/5G	POMVADE	110,00DA	03/2014	18
<input type="checkbox"/>	611000C90109	COBAVINE 100J/EG/ML B/5	INJECTION	220,00DA	07/2013	19
<input type="checkbox"/>	611000C90110	COVERSYL 4MG B/30	COMPRIME	1463,38DA	12/2014	10
<input type="checkbox"/>	611000C90111	COZAAR 100MG B/28	COMPRIME	1 275,37DA	01/2013	11
<input type="checkbox"/>	611000C90112	DICLAMID 5MG/ML BT/2	INJECTION	110,06DA	01/2016	12
<input type="checkbox"/>	611000C90113	DILACARD 6,25MG BTE/30	COMPRIME	696,22DA	11/2015	13
<input type="checkbox"/>	611000C90114	DRIPTAVE 5MG BTE/6	COMPRIME	732,37DA	12/2013	14
<input type="checkbox"/>	611000C90115	GLUCOBAY 50MG B/30	COMPRIME	526,92DA	12/2013	15
<input type="checkbox"/>	611000C90116	GYLAXA 5MG BTE/20	COMPRIME	249,62DA	10/2013	16
<input type="checkbox"/>	611000C90117	IMINOPRIL 10MG B/28	COMPRIME	901,04DA	01/2015	17

Figure 4 : La table pharmacie

III.3 Accéder à notre base de données à partir de NetBeans :

- ✦ Lancer NetBeans IDE 6.8, cliquer sur « *Services* » et choisir « *Database* » → « *Drivers* »
Voir figure 5.

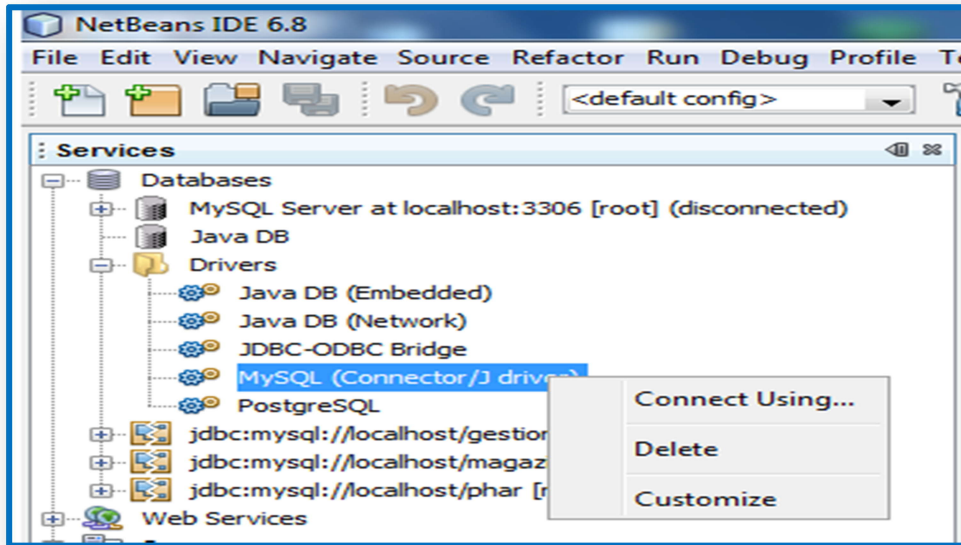


Figure 5 : Connecter a la base de données

- ✚ Cliquez-droit sur « **MySQL (Connector/J driver)** » → « **ConnectUsing ...** » Saisir le nom de la base de données dans le champ « **Database** », cliquez sur « **Ok** ».

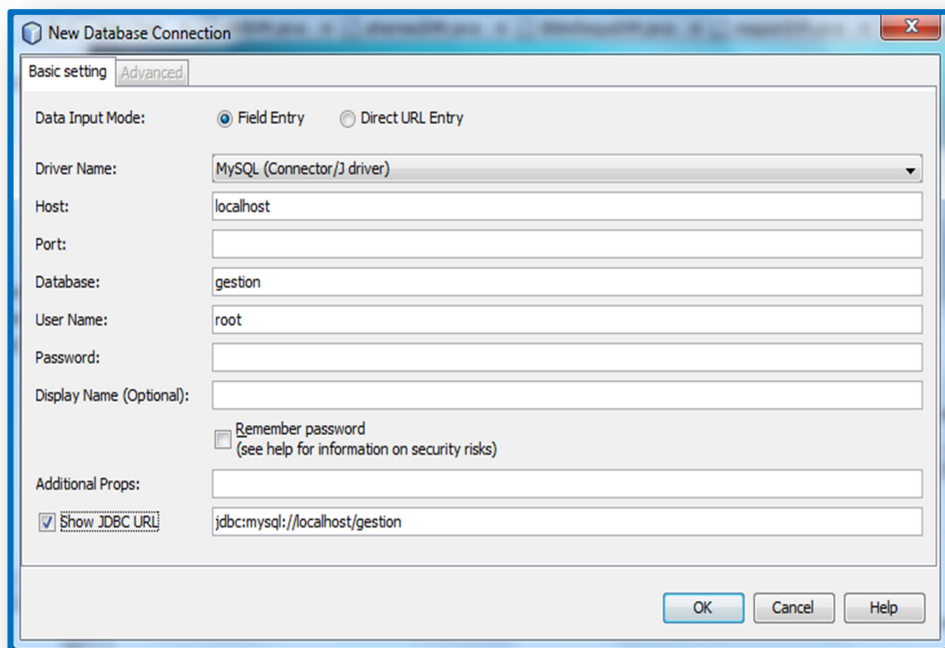


Figure 6: connecter à la base de données

- ✚ Notre base de données est maintenant connectée, pour afficher la table cliquez-droit Sur « **gestion** » en suite choisir « **View Data ...** »

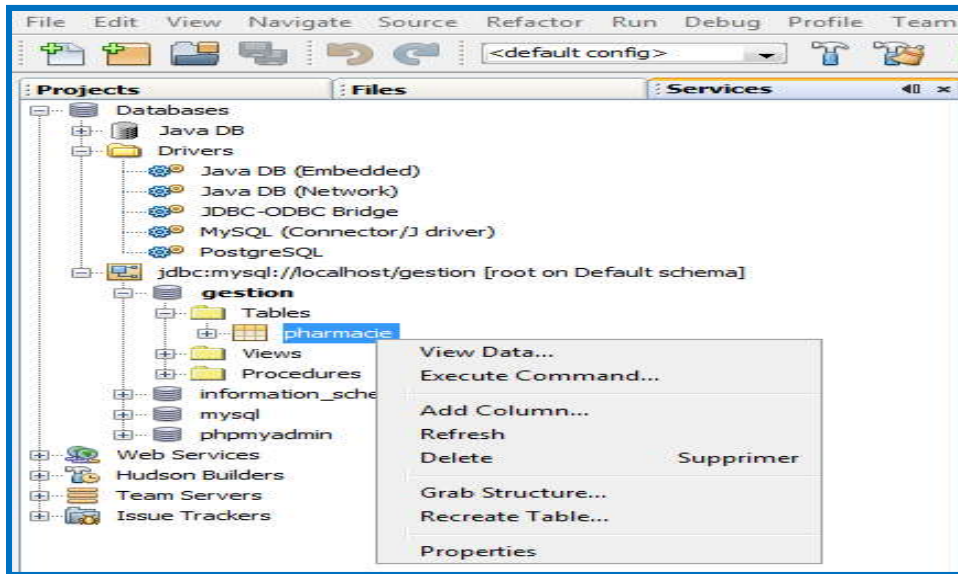


Figure 7 : Affichage de la table sous NetBeans

- ✦ S'affiche ensuite

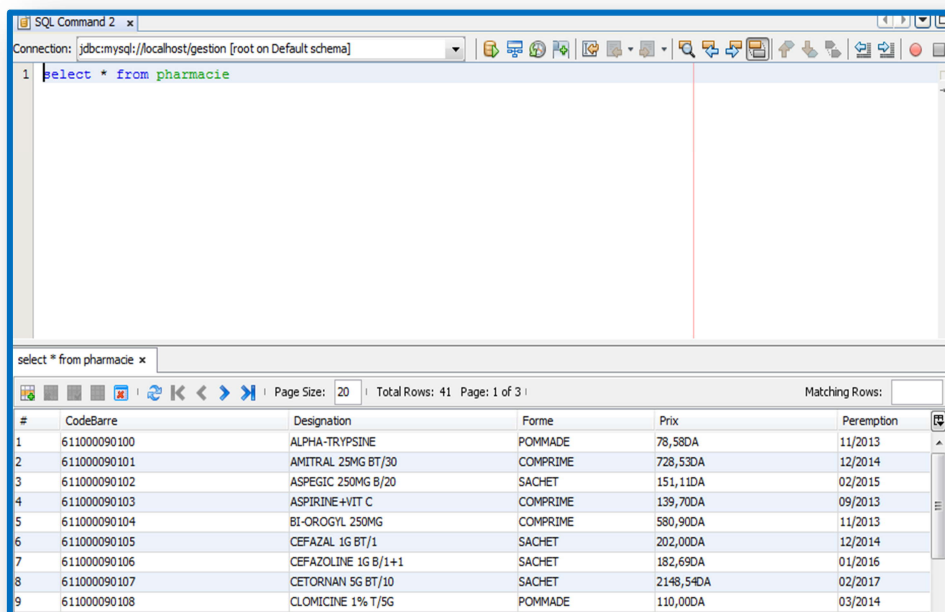


Figure8 : Résultat de l'affichage de la table

III.4 Création du projet :

III.4.1 Créer un projet Swing :

- ✦ Cliquez sur « **File** » → « **New Project** », choisir « **Java Application** » de catégories « **Java** » et cliquez sur « **Next** »
- ✦ Saisir le nom du projet, pour notre projet c'est « **gestion** », désélectionner

✚ « *Create Main Class* », cliquez sur « *Finish* ». voir **Figure 9**

III.4.2 L'ajout du framework « Hibernate » dans la librairie

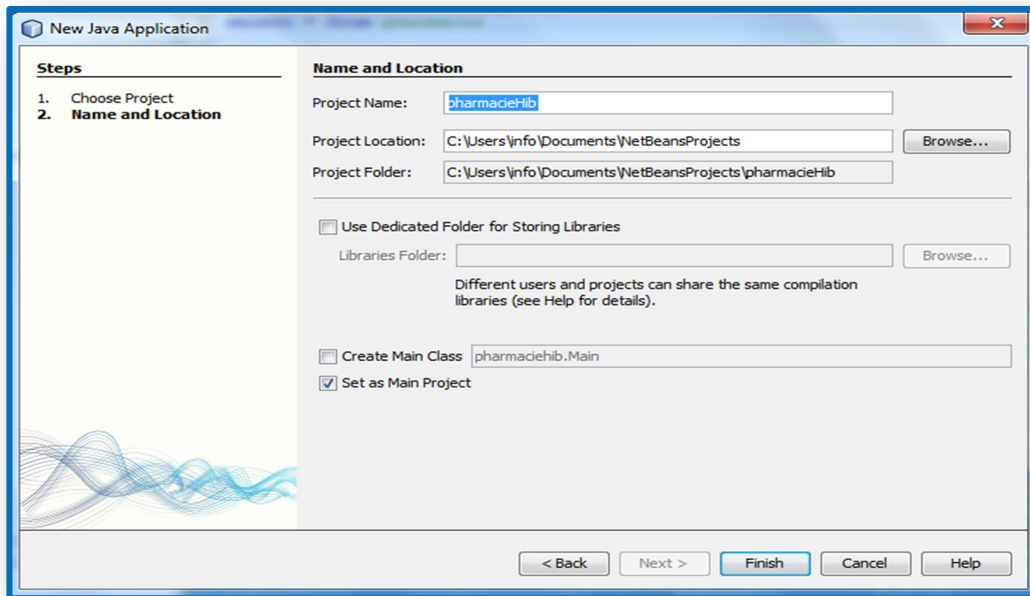


Figure 9 : Créer le projet

✚ Cliquez-droit sur « **libraries** » et appuyer sur « **Add Library** » et choisir « **Hibernate** »

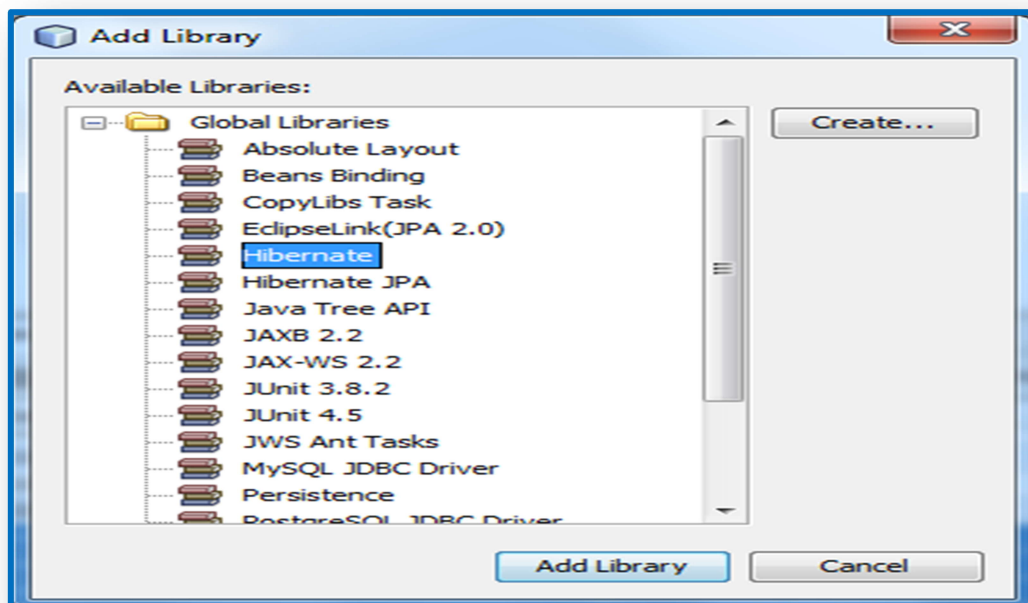


Figure 10: Ajout de la librairie Hibernate

III.4.3 Arborescence du projet

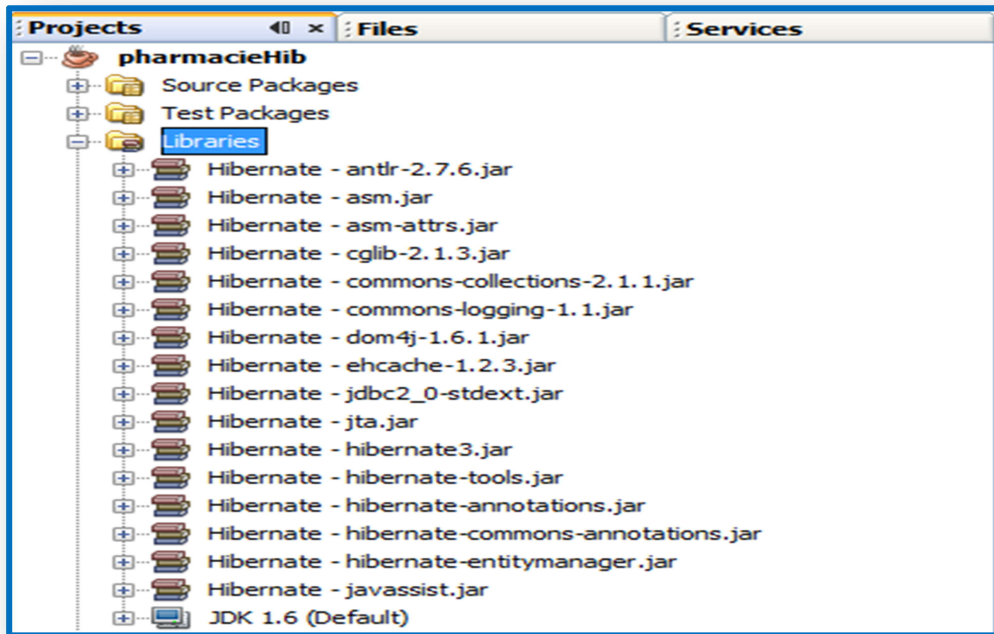


Figure 11 : Arborescence du projet

III.4.4 Créer le fichier de configuration hibernate.cfg.xml :

- Cliquez sur « *Source Packages* », choisir « *New* » → « *Other* » .
- Sélectionnez « *Hibernate Configuration Wizard* » de catégories « *Hibernate* »

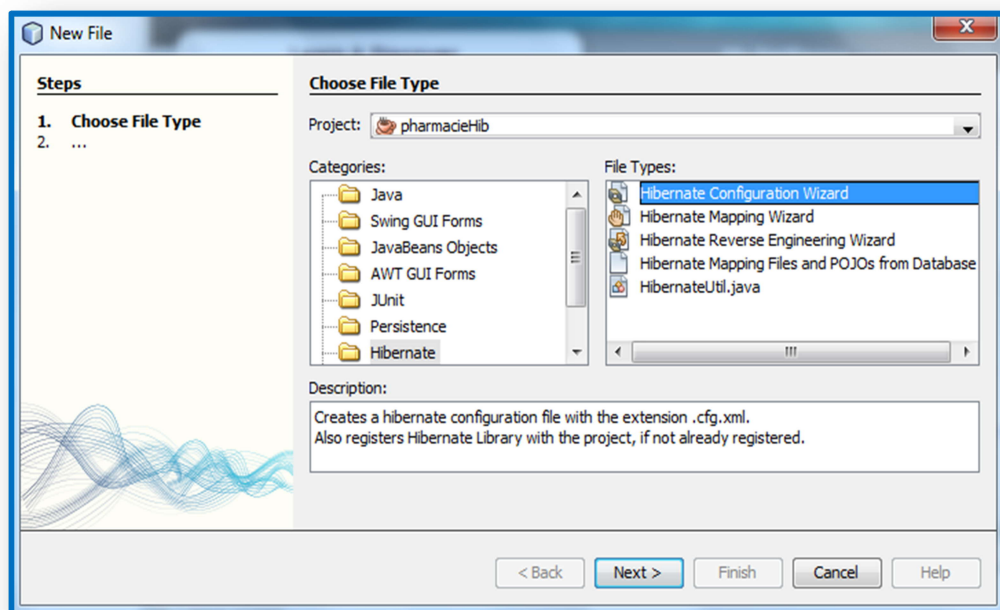


Figure 12 : Créer le fichier de configuration(1/3)

- ✦ Cliquez sur « *next* » .
- ✦ S'affiche :

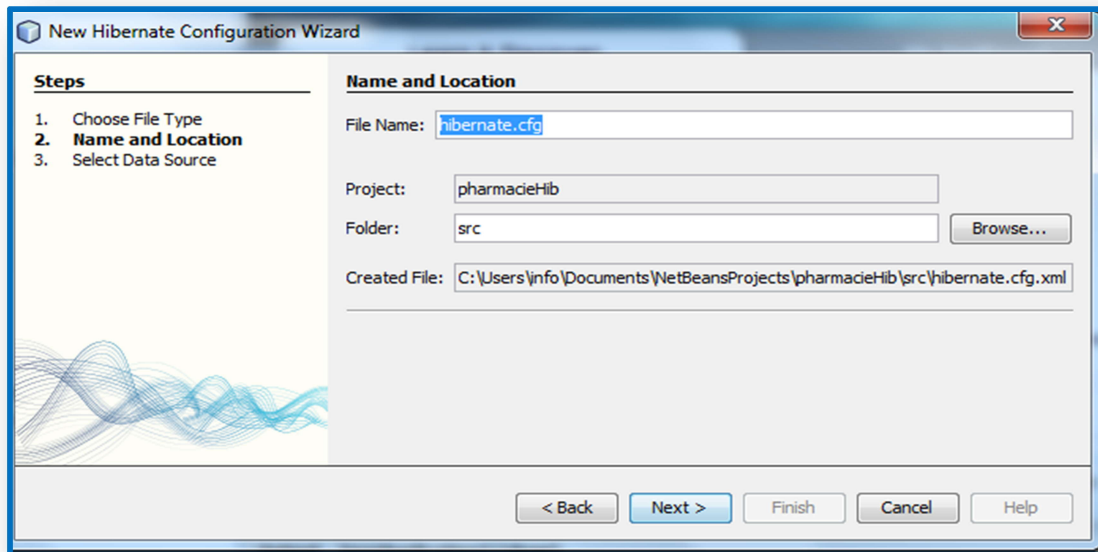


Figure 13 : Créer le fichier de configuration (2/3)

- ✦ Cliquez sur « *next* » .
- ✦ Sélectionnez la base de données « *gestion* ».
- ✦ Cliquez sur « *Finish* » .

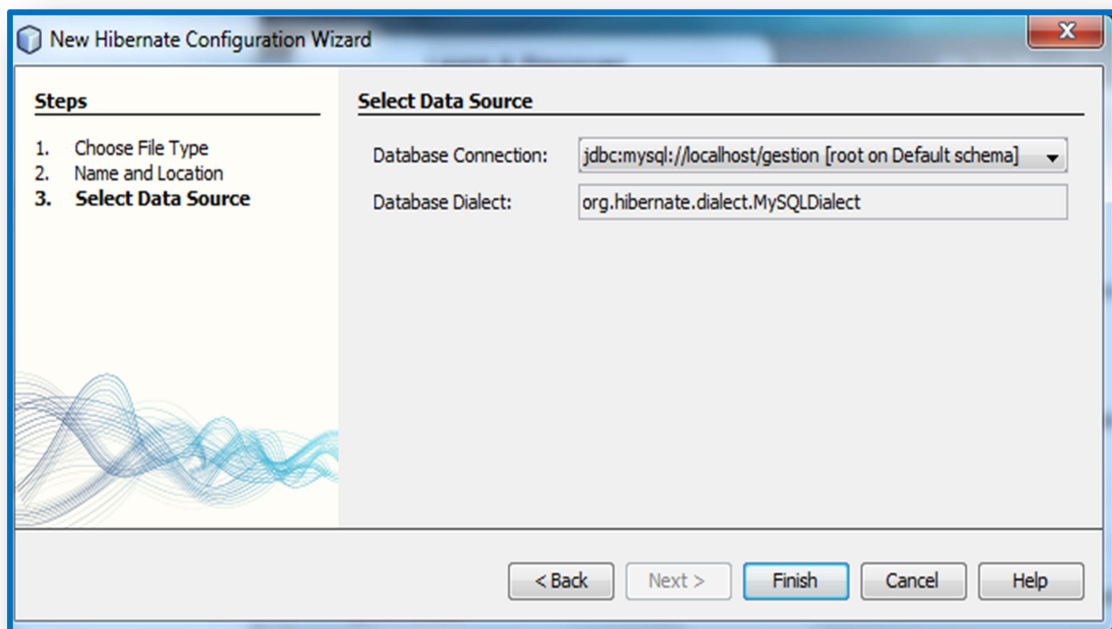


Figure 14 : Créer le fichier de configuration (3/3)

- ✦ S'affiche : hibernate.cfg.xml : contenu *Design*

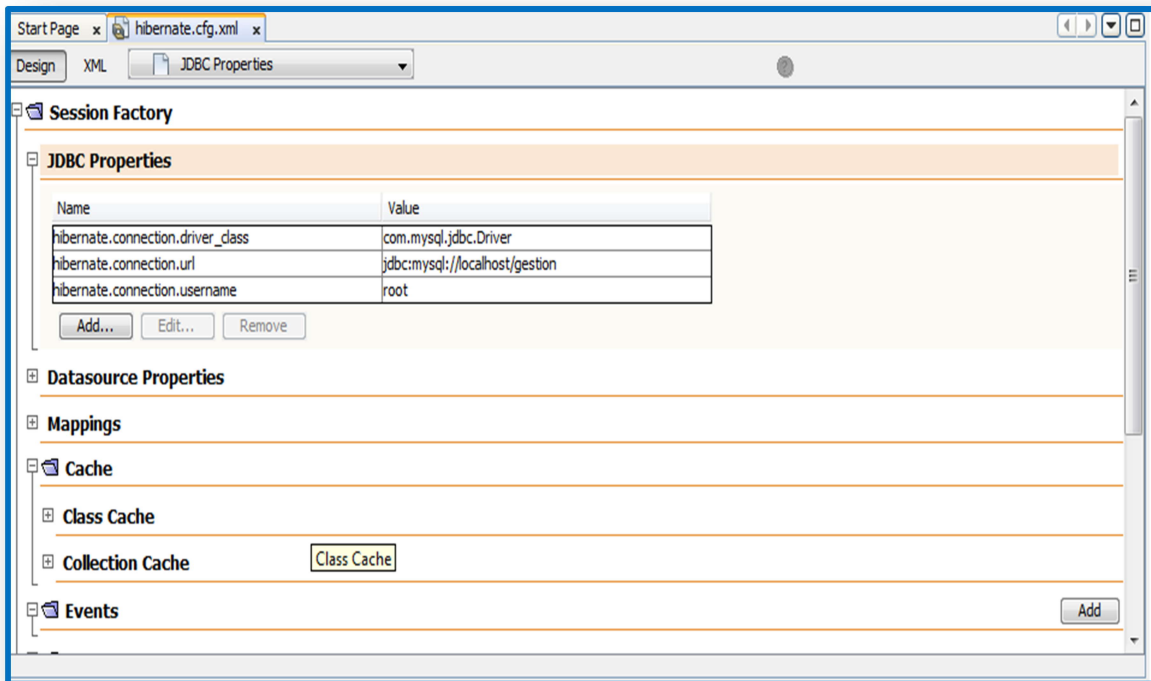


Figure 15 : Contenu design de fichier de configuration

- ✦ Hibernate.cfg.xml : Contenu *XML*

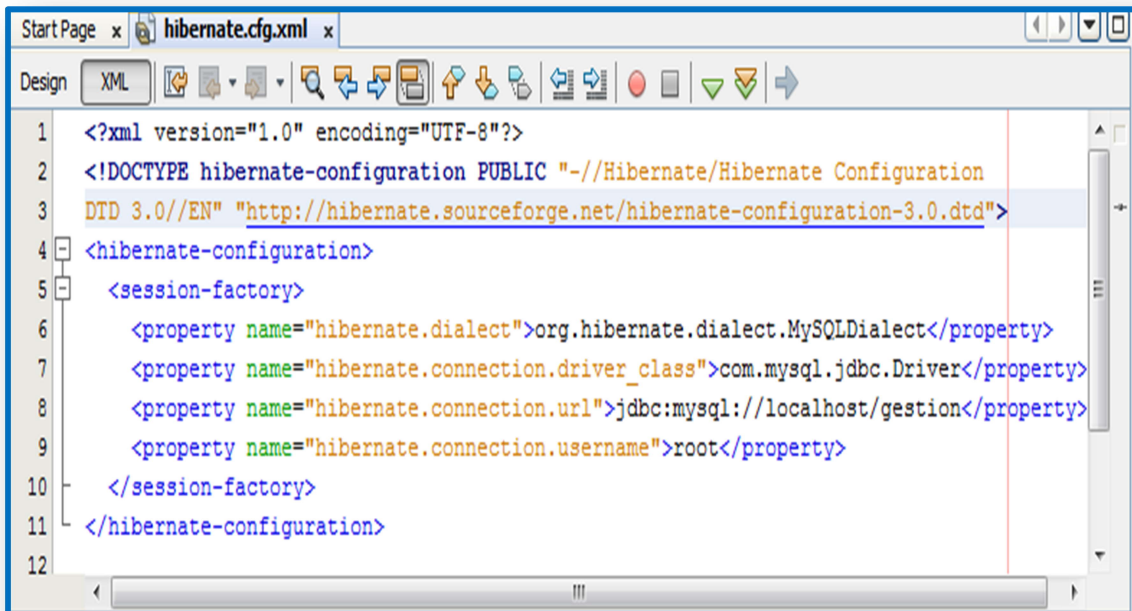


Figure 16 : Contenu XML de fichier de configuration

III.4.5 Modifier hibernate.cfg.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property name="hibernate.dialect">org.hibernate.dialect.MySQL Dialect</property>
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost/badr</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">badr</property>
<property name="hibernate.show_sql">>true</property>
<property
name="hibernate.query.factory_class">org.hibernate.hql.classic.ClassicQueryTranslatorFac
tory</property>
</session-factory>
</hibernate-configuration>
```

- ▀ **hibernate.show_sql (true)** : permet d'afficher les requêtes dans la console de l'IDE.
- ▀ **hibernate.query.factory_class** : choisi l'implémentation du parseur de requête .

III.4.6 Créer le « Helper file » : fichier HibernateUtil .java :

Clique-droit sur « *Source Packages* », Sélectionnez « *New* » → *other* ».

Sélectionnez « *Hibernate* » de catégories et « *HibernateUtil.java* » de File types .

Cliquez « *Next* ».

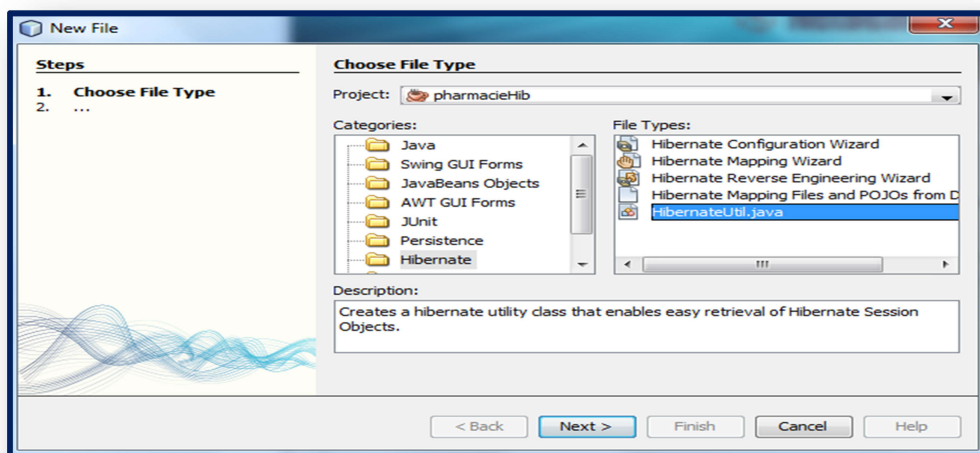


Figure 17 : Créer le Helper File (1/2)

- ✦ saisir « *hibernateUtil* » pour le nom de classe, saisir « *pharmacie.util* » comme nom de package, cliquez sur « *Finish* ».

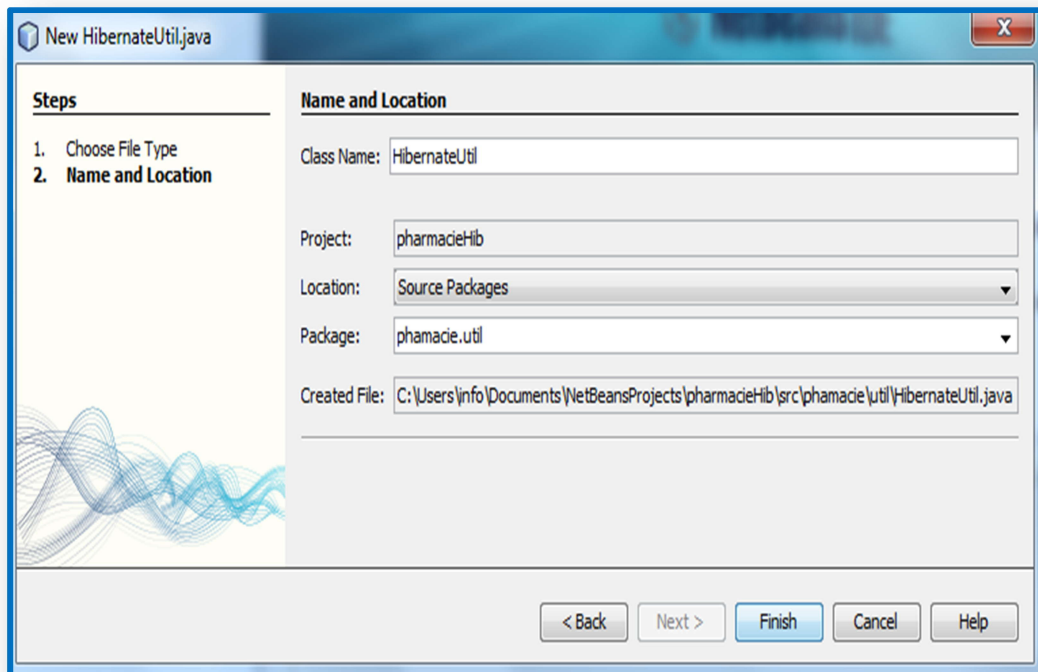


Figure 18 : Créer le Helper File (2/2)

- ✦ Ci-dessous la description du « Helper » :

```
package pharmacie.util;
import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;
public class HibernateUtil {
private static final SessionFactory sessionFactory;
static {
try {
sessionFactory = new
AnnotationConfiguration().configure().buildSessionFactory();
} catch (Throwable ex) {
System.err.println ("Initial SessionFactory creation failed." + ex);
throw new ExceptionInInitializerError(ex);
}
}
public static SessionFactory getSessionFactory() {
return sessionFactory; }
}
```

III.4.7 Générer les fichiers de mapping et de classes java :

- ✦ Pour mapper « pharmacie.java » à la table « pharmacie » , on doit passer par deux étapes :
- Hibernate Reverse Engineering Wizard (.xml).
- Hibernate Mapping Files (.xml) and POJOs from a Database (.java)

III.4.8 Créer le fichier « Reverse Engineering » :

- ✦ Cliquez-droit sur « *Source Packages* », sélectionnez « *New* » → « *Other* ».
- ✦ Sélectionnez « *Hibernate* » de catégories et « *Hibernate reverse Engineering Wizard* » de File Types.
- ✦ Cliquez sur « *Next* ».

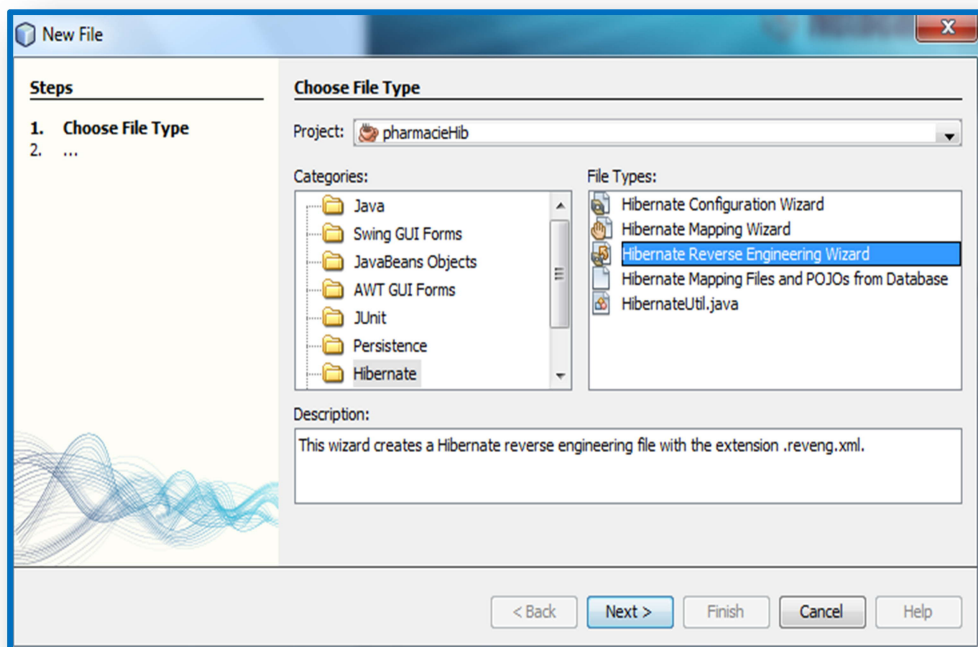


Figure 19 : Créer le fichier Reverse Engineering (1/2)

- ✦ Ensuite :
- ✦ Saisir « *Hibernate.reveng* » comme nom.
- ✦ Laissez « *Src* » comme Folder.

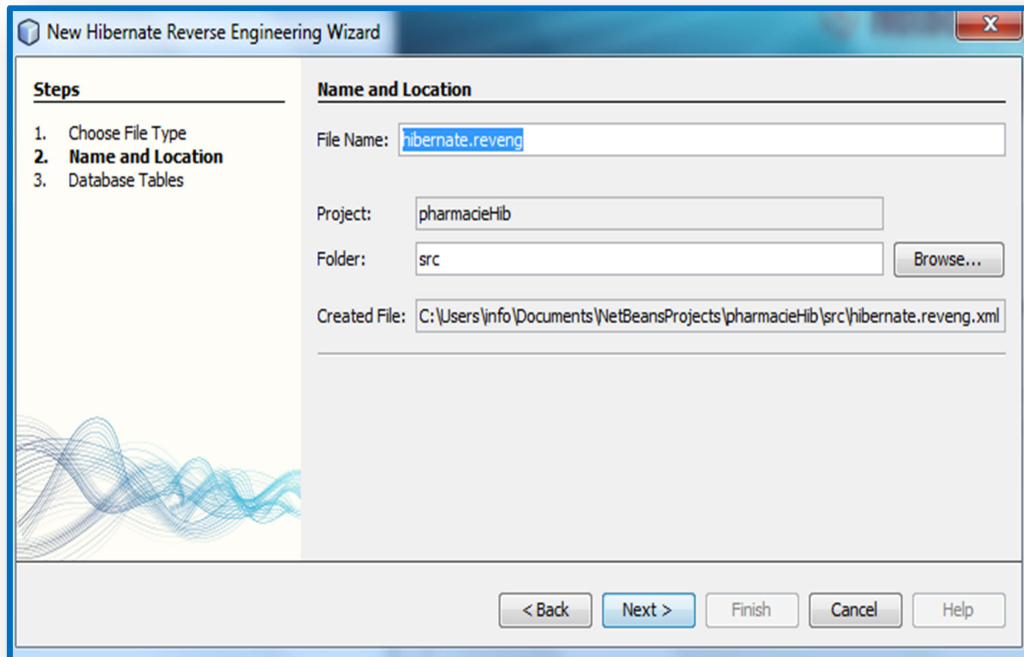


Figure 20 : Créer le fichier Reverse Engineering (2/2)

⬇ S'affiche :

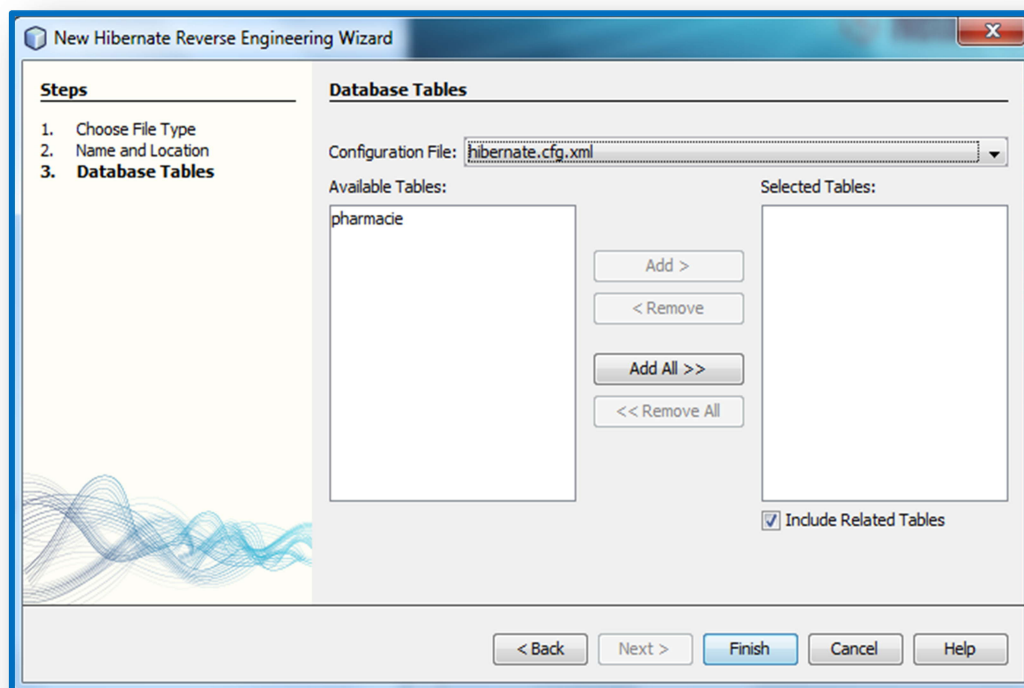


Figure 21: Sélection de la table pharmacie

⬇ Sélectionnez la table « *pharmacie* », choisir « *Add* » et cliquez sur « *Finish* ».

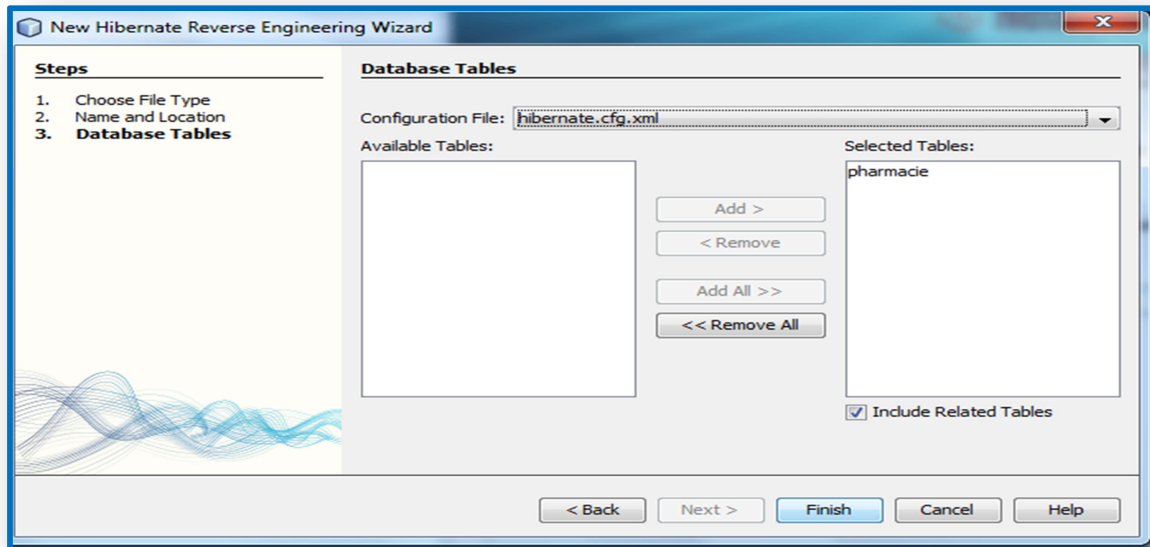


Figure 22 : Ajouter la table pharmacie

✚ Ci-dessous le contenu du hibernate.reveng.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse
Engineering DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-reverse-
engineering-3.0.dtd">
<hibernate-reverse-engineering>
<schema-selection match-catalog="gestion"/>
<table-filter match-name="pharmacie"/>
</hibernate-reverse-engineering>
<
```

III.4.9 Créer «Hibernate Mapping Files and POJOs From a Database :

- ✚ Cliquez sur « *Source Packages* », sélectionnez « *New* » → « *Other* ».
- ✚ Sélectionnez « *Hibernate* » de catégories et « *Mapping Files and POJOs from a Database* » de File Types.
- ✚ Cliquez sur « *Next* ».

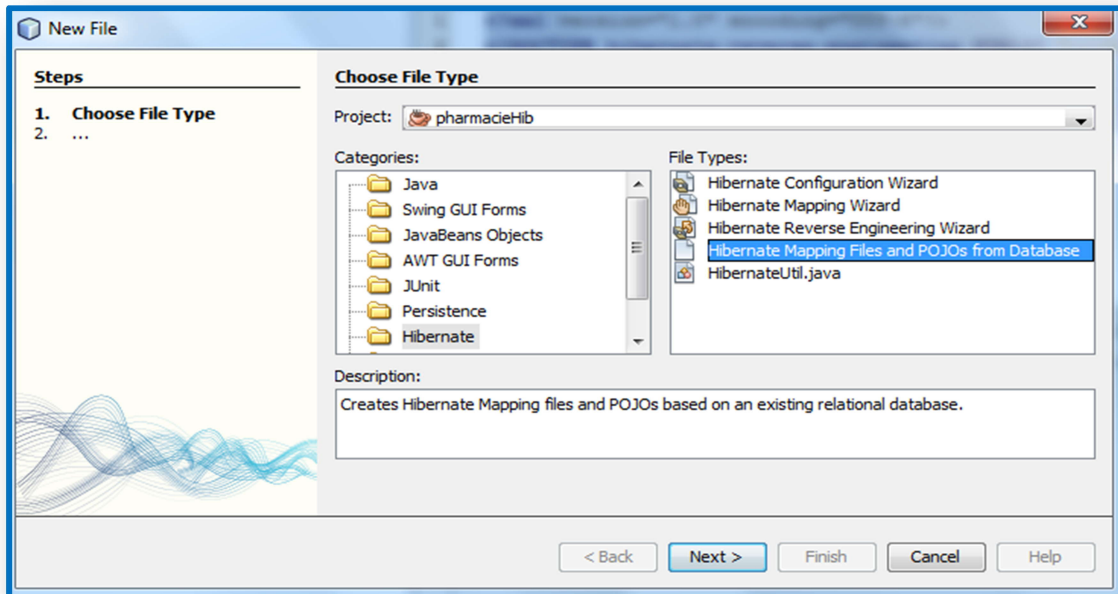


Figure 23: Hibernate Mapping Files and POJOs from a Database (1/2)

- ✦ Sélectionnez « *hibernate.cfg.xml* » de « *Hibernate Configuration File* ».
- ✦ Sélectionnez « *hibernate.reveng.xml* » de « *Hibernale Reverse Engineering File* ».
- ✦ Assurer que « *Domain Code* » et « *Hibernate XML Mappings* » sont cochés.
- ✦ Saisir « *pharmacie.entity* » comme nom du package, cliquer « *Finish* ».

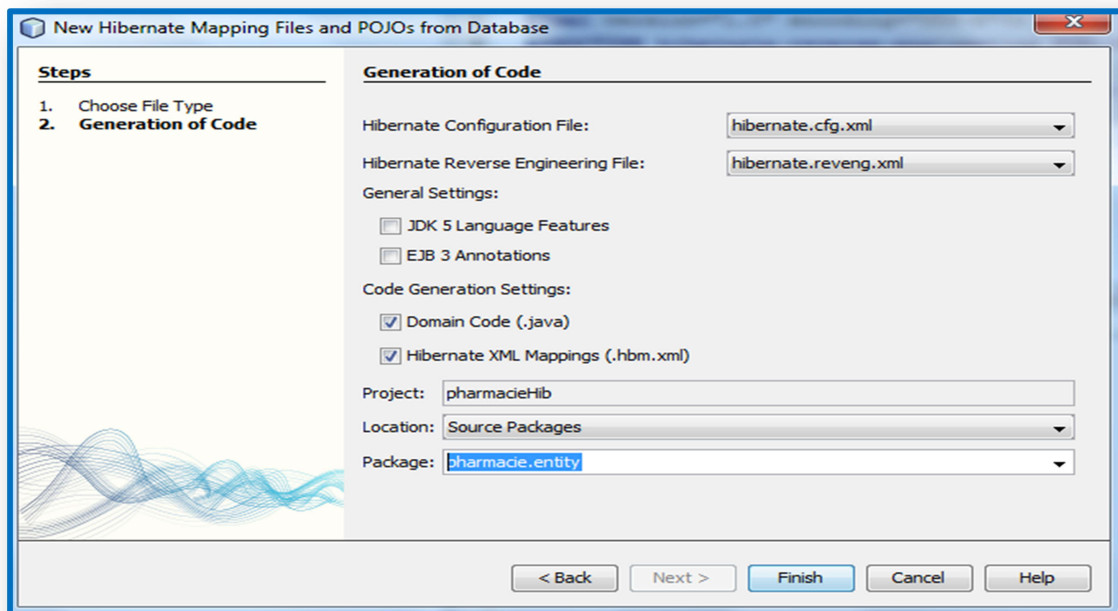


Figure 24: Hibernate Mapping Files and POJOs from a Database (2/2)

- ✦ Ci-dessous l'arborescence de notre projet:

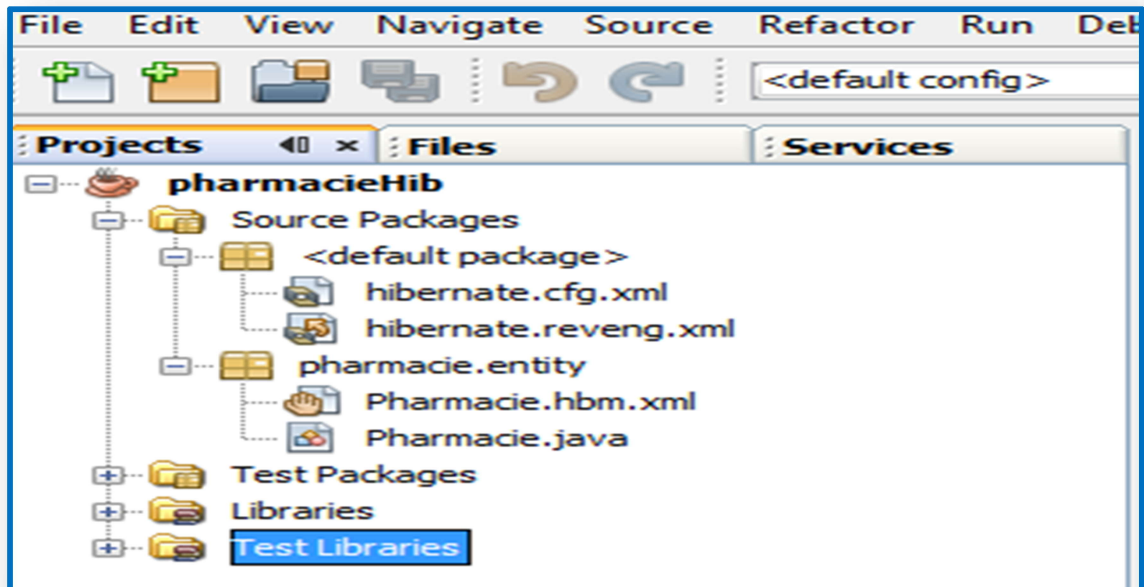



Figure 25 : Arborescence du projet

↳ Pharmacie.java(POJO) :

```
package pharmacie.entity;
public class Pharmacie implements java.io.Serializable {
private Int code Barre;
private String designation;
private String forme;
private String prix;
private String peremption;
private int stock
public Pharmacie() {
}
public Pharmacie(intcodeBarre, String designation, String forme, String prix, String
peremption, int stock) {
this.codeBarre = codeBarre
this.designation = designation;
this.forme = forme;
this.prix = prix;
this.peremption = peremption;
this.stock = stock;
}
public String getCodeBarre() {
```

```
        return this.codeBarre;
    }
    public void setCodeBarre(String codeBarre) {
        this.codeBarre = codeBarre;
    }
    public String getDesignation()
        return this.designation;
    }
    public void setDesignation(String designation) {
        this.designation = designation; }
    public String getForme() {
        return this.forme;
    }
    public void setForme(String forme) {
        this.forme = forme;
    }
    public String getPrix() {
        return this.prix;
    }
    public void setPrix(String prix) {
        this.prix = prix;
    }
    public String getPeremption() {
        return this.peremption;
    }
    public void setPeremption(String peremption) {
        this.peremption = peremption;
    }
    public int getStock() {
        return this.stock;
    }
    public void setStock(int stock) {
        this.stock = stock;
    }
}
```

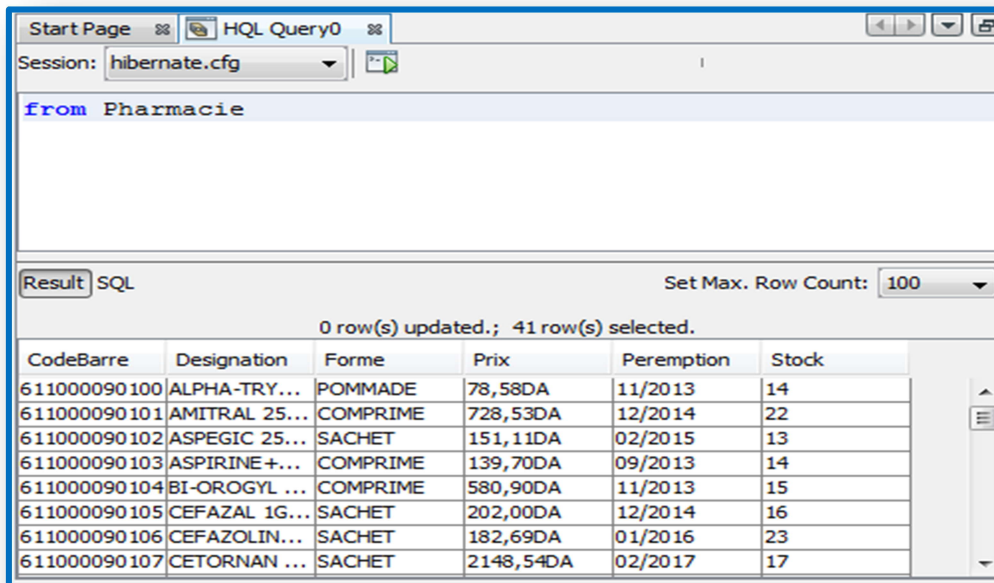
 Pharmacie.hbm.xml (le fichier de mapping) :

```
<?xml version="1.0"?><!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated 10 mai 2013 16:37:45 by Hibernate Tools 3.2.1.GA -->
<hibernate-mapping>
<class name="pharmacie.entity.Pharmacie" table="pharmacie" catalog="gestion">
<id name="codeBarre" type="string">
<column name="CodeBarre" length="40" />
<generator class="assigned" />
</id>
<property name="designation" type="string">
<column name="Designation" length="40" not-null="true" />
</property>
<property name="forme" type="string">
<column name="Forme" length="15" not-null="true" /></property>
<property name="prix" type="string">
<column name="Prix" length="20" not-null="true" /></property>
<property name="peremption" type="string">
<column name="Peremption" length="20" not-null="true" /></property>
<property name="stock" type="int">
<column name="Stock" not-null="true" />
</property>
</class>
</hibernate-mapping>
```

🔧 Changement dans hibernate.cfg.xml :

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration><session-factory>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost/gestion</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.show_sql">>true</property>
<propertyname="hibernate.query.factory_class">
org.hibernate.hql.classic.ClassicQueryTranslatorFactory</property>
<mapping resource="pharmaci/entity/Pharmacie.hbm.xml"/></session-factory>
</hibernate-configuration>
```

- ✦ Faire des tests avec des requêtes HQL :



CodeBarre	Designation	Forme	Prix	Peremption	Stock
611000090100	ALPHA-TRY...	POMMADE	78,58DA	11/2013	14
611000090101	AMITRAL 25...	COMPRIME	728,53DA	12/2014	22
611000090102	ASPEGIC 25...	SACHET	151,11DA	02/2015	13
611000090103	ASPIRINE+...	COMPRIME	139,70DA	09/2013	14
611000090104	BI-OROGYL ...	COMPRIME	580,90DA	11/2013	15
611000090105	CEFAZAL 1G...	SACHET	202,00DA	12/2014	16
611000090106	CEFAZOLIN...	SACHET	182,69DA	01/2016	23
611000090107	CETORNAN ...	SACHET	2148,54DA	02/2017	17

Figure 26: Test avec requête HQL

III.5 Créer une JFrame pour faire des recherches dans la table des produits d'une pharmacie :

III.5.1 Créer la JFrame :

- ✦ Cliquez-droit sur le projet, choisir « *New* » → « *Other* » → sélectionnez « *JFrameFrom* » de la categories « *Swing GUI Forms* », cliquez sur « *Next* ».

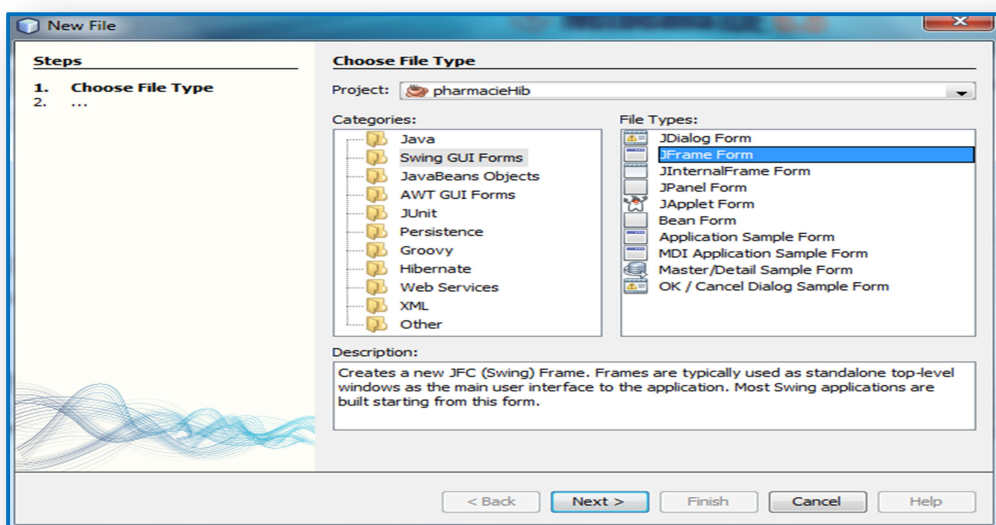


Figure 27 : Créer la JFrame (2/2)

Chapitre 2 : Présentation de l'application

- Ensuite, saisi « **pharmacieIHM** » comme nom de la classe et « **pharmacie.ui** » pour le package, cliquez sur « **Finish** ».

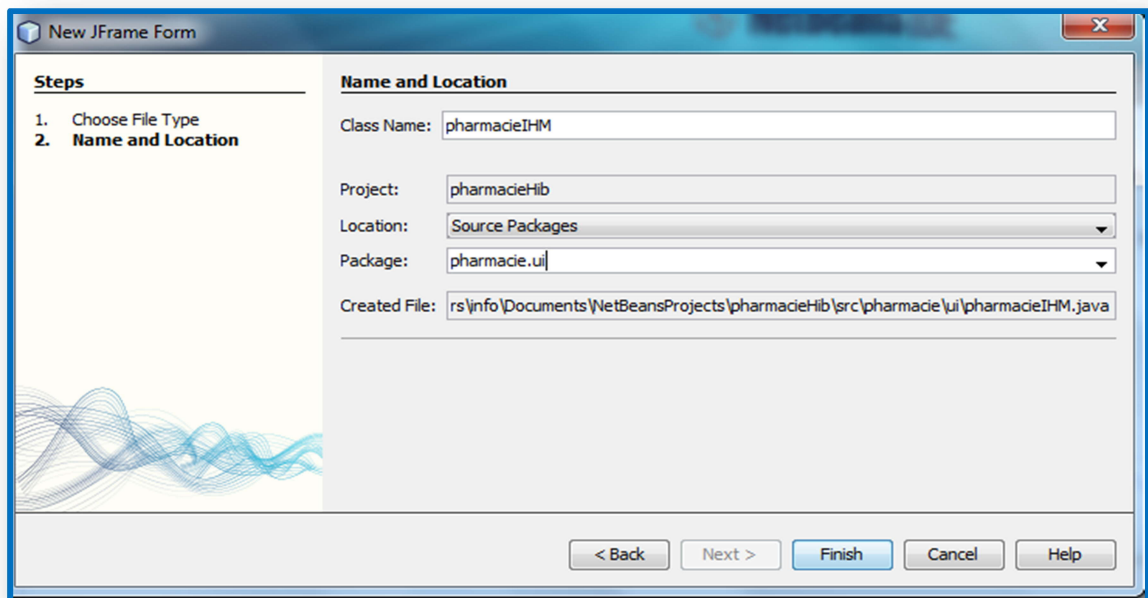


Figure 28 : Créer la JFrame (2/2)

- S'affiche :

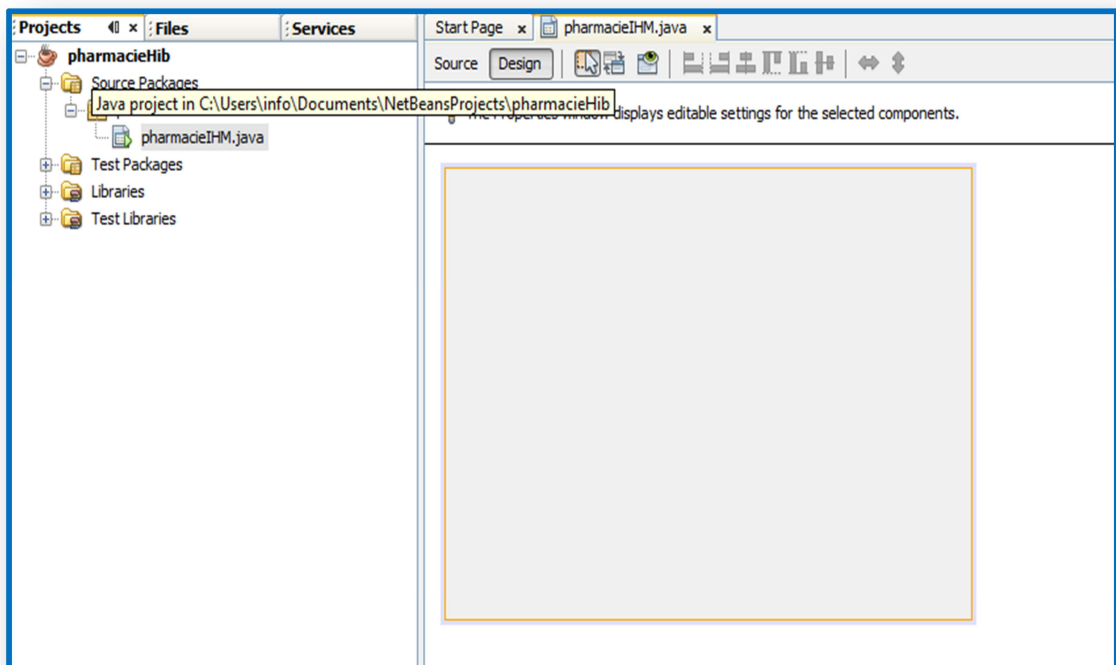


Figure 29 : Zone de la JFrame

III.5.2 Ajouter des éléments a la JFrame :

- Cliquez-droit sur la forme graphique :
 - Add Form Palette → Swing Containers → Panel
 - Add Form palette → Swing Menus → Menu Bar
 - Ajouter des menus au menu Bar
 - Add Form palette → Swing Controls → Radio Button (3 Radio Button)
- Saisit : « Recherche par code barre »
- Saisit : « Recherche par désignation »
- Saisit : « Recherche par prix »
 - Add Form palette → Swing Controls → Button Group
 - Add Form palette → Swing Controls → Text Field
- Supprimer le texte par défaut.
- Add Form palette → Swing Controls → Button (2Button)
- Modifier le texte à « Afficher ».
- Modifier le texte à « Quitter ».
- Add Form palette → Swing Controls → Table
- Cliquez-droit sur la table → Table → Contents → Columns → Count(6)

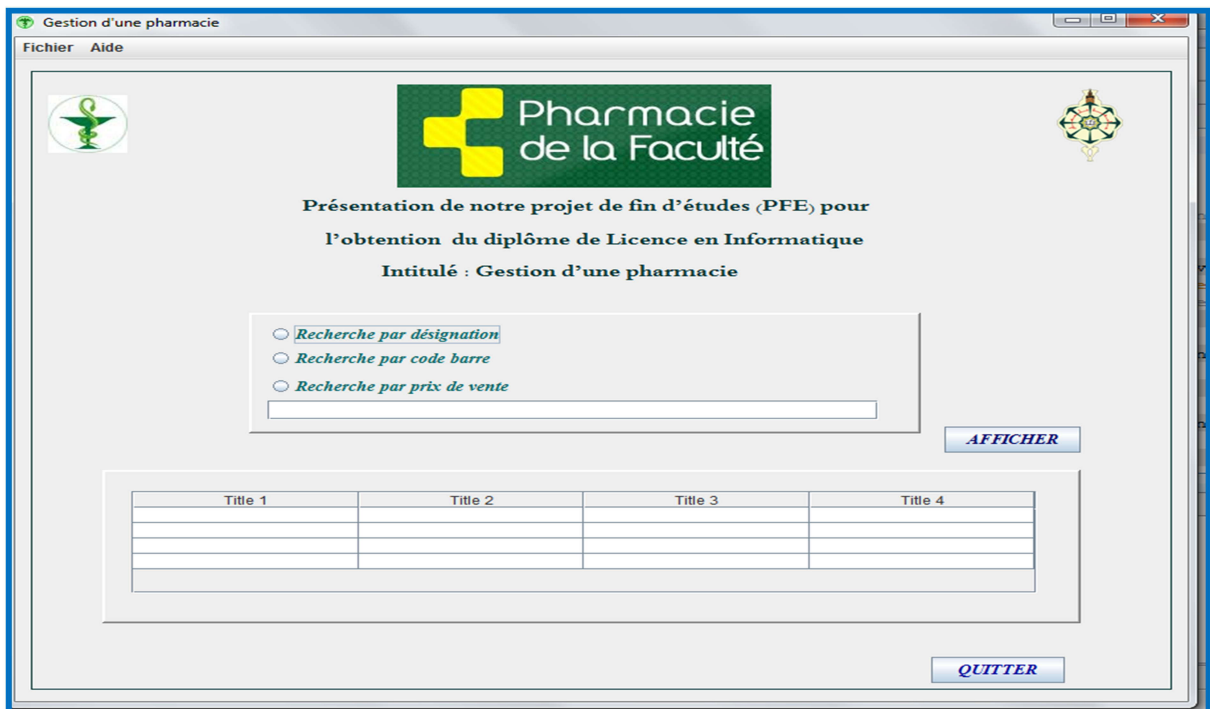


Figure 30 : Interface principale

III.5.3 Changeant le nom des variables :

✚ Pour Chaque élément :

- Clique-droit → « Change Variable Name »
- La table → resultTable
- La zone de texte → champ

III.5.4 Ajouter les déclarations suivantes au code source :

```
private static String QUERY_BASED_ON_NAME1=" select from Pharmacie a where
a.codeBarre like ";

private static String QUERY_BASED_ON_NAME2=" select from Pharmacie b where
b.designation like ";

private static String QUERY_BASED_ON_NAME3=" select from Pharmacie c where c.prix
like ";
```

III.5.5 Ajouter les méthodes suivantes, qui traitent le caractère saisi à partir de l'IHM :

```
private void rechercherparcodebarre(){
executeHQLQuery(QUERY_BASED_ON_NAME1+ champ.getText() + "%");}
private void rechercherpardesignation(){
executeHQLQuery(QUERY_BASED_ON_NAME2+ champ.getText() + "%");}
private void rechercherparprix(){
executeHQLQuery(QUERY_BASED_ON_NAME3 + champ.getText() + "%");}
```

III.5.6 Ajouter la méthode executeHQLQuery() :

```
private void executeHQLQuery(String hql) {
try {
    Session session = HibernateUtil.getSessionFactory().openSession();
    session.beginTransaction(); //Démarrer la transaction
    Query q = session.createQuery(hql);
    List resultList = q.list(); // Exécuter la requête et récupérer le résultat sous forme de List
    displayResult(resultList);
    session.getTransaction().commit(); //Récupérer la transaction et la valider
} catch (HibernateException he) {
    he.printStackTrace(); }}
```

III.5.7 Ajouter les imports nécessaires:

- `import pharmaci.util.HibernateUtil;`
- `import java.util.List;`
- `import org.hibernate.HibernateException;`
- `import org.hibernate.Query;`
- `import org.hibernate.Session;`

III.5.8 Basculer vers Design et double clic sur le bouton Recherche :

✚ Modifier la méthode `queryButtonActionPerformed`

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    if (!champ.getText().trim().equals("")) {
        if(jRadioButton1.isSelected()) {
rechercheparcodebarre(); }
        else if (jRadioButton2.isSelected()) {
recherchepardesignation() ; }
        else if (jRadioButton3.isSelected()) {
rechercheparprix() ; }
    else
JOptionPane.showMessageDialog(null, "svp choisissez un mode de recherche"); }
    else
JOptionPane.showMessageDialog(null, "svp donnez un caractère");}
```

III.5.9 Ajouter la méthode `displayresult` :

```
private void displayResult(List resultList) {
    Vector<String>tableHeaders = new Vector<String>(); // construire l'entête de la table
    VectortableData = new Vector(); // récupérer les données ligne par ligne
    tableHeaders.add("CodeBarre");
    tableHeaders.add("Designation");
    tableHeaders.add("Forme");
    tableHeaders.add("Prix");
    tableHeaders.add("Peremption");
    tableHeaders.add("Stock");
    for (Object o : resultList) {
```

```
Pharmacie f = (Pharmacie) o;  
    Vector<Object>oneRow = new Vector<Object>();  
oneRow.add(f.getCodeBarre());  
oneRow.add(f.getDesignation());  
oneRow.add(f.getForme());  
oneRow.add(f.getPrix());  
oneRow.add(f.getPeremption());  
oneRow.add(f.getStock());  
tableData.add(oneRow); }  
resultTable.setModel(new DefaultTableModel(tableData, tableHeaders));}
```

III.5.10 Ajouter les imports nécessaires :

- `import java.util.Vector;`
- `import javax.swing.JOptionPane;`
- `import javax.swing.table.DefaultTableModel;`
- `import pharmacie.entity.Pharmacie;`

III.6 Créer le JDialog pour faire la recherche par forme et la recherche par péremption :

III.6.1 Créer et ajouter des éléments a la JDialog :

(nous allons montrer dans ce qui va suivre les étapes nécessaires pour « la recherche par forme » seulement car elles sont les mêmes que pour « la recherche par date de péremption »).

- Cliquez-droit sur le projet, choisir « *New* » → « *Other* », sélectionnez « *JDialogFrom* » de la catégorie « *Swing GUI Forms* », cliquez sur « *Next* ».

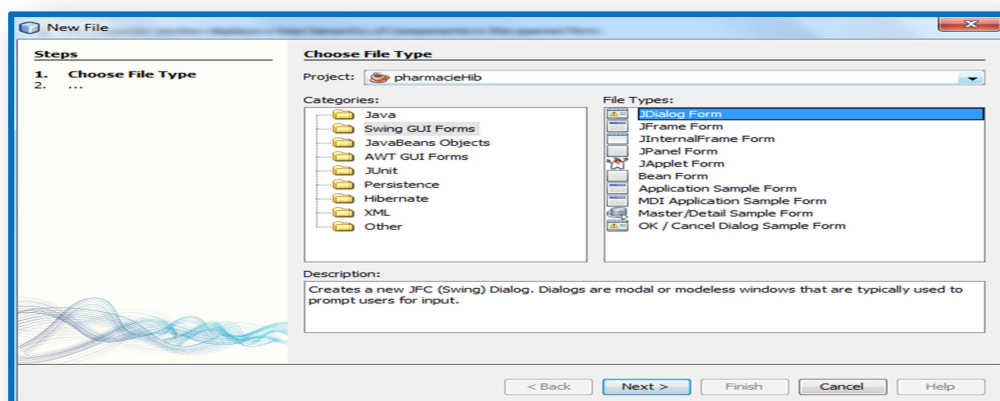


Figure 31: créer la JDialog (1/2)

- Ensuite, saisi « *Rechercheparforme* » comme nom de la classe et « *pharmacie.ui* » pour le package, cliquez sur « *Finish* ».

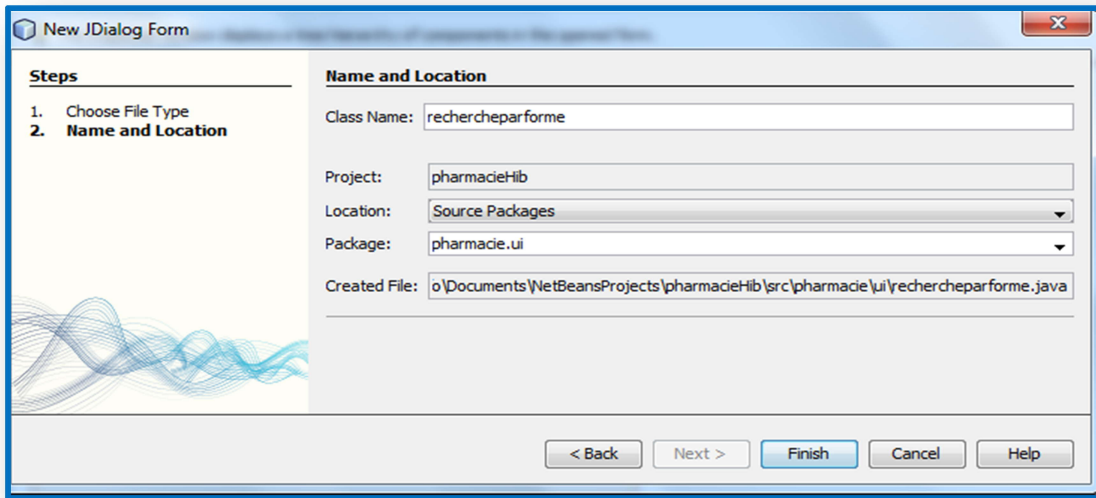


Figure 32 : créer la JDialog (2/2)

- S'affiche

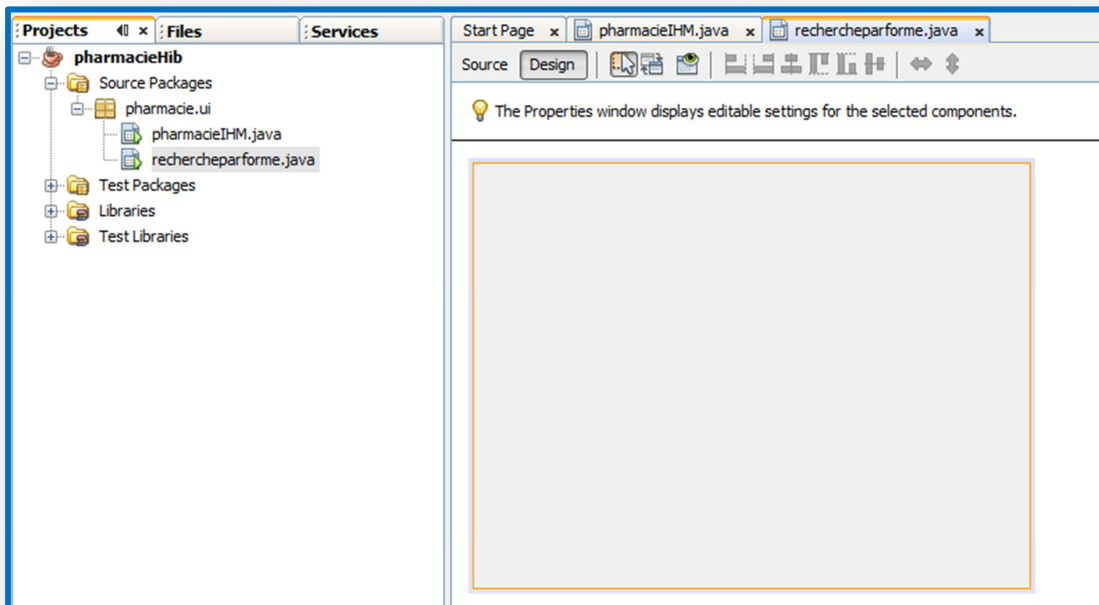


Figure 33 : Zone de JDialog

- Cliquez-droit sur la forme graphique :
 - Add Form Palette → Swing Containers → Panel
 - AddForm palette → Swing Controls → Combo Box (pour la recherche par forme).
 - Modifier le texte.

Chapitre 2 : Présentation de l'application

- Add Form palette → Swing Controls → Textfield (pour la recherche par date de péremption)
- Add Form palette → Swing Controls → Button (2 Button)
 - Modifier le texte à « Afficher ».
 - Modifier le texte à « Fermer »
- Add Form palette → Swing Controls → Table
Clique-droit sur la table → Table Contents → Columns → Count(6)

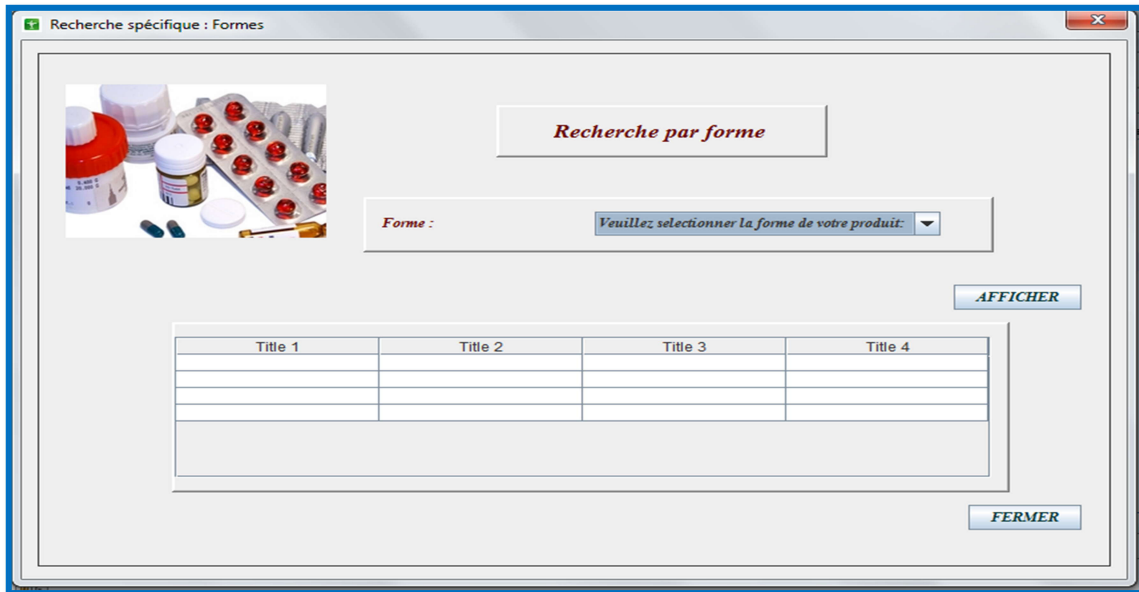


Figure 34 : Interface recherche par forme

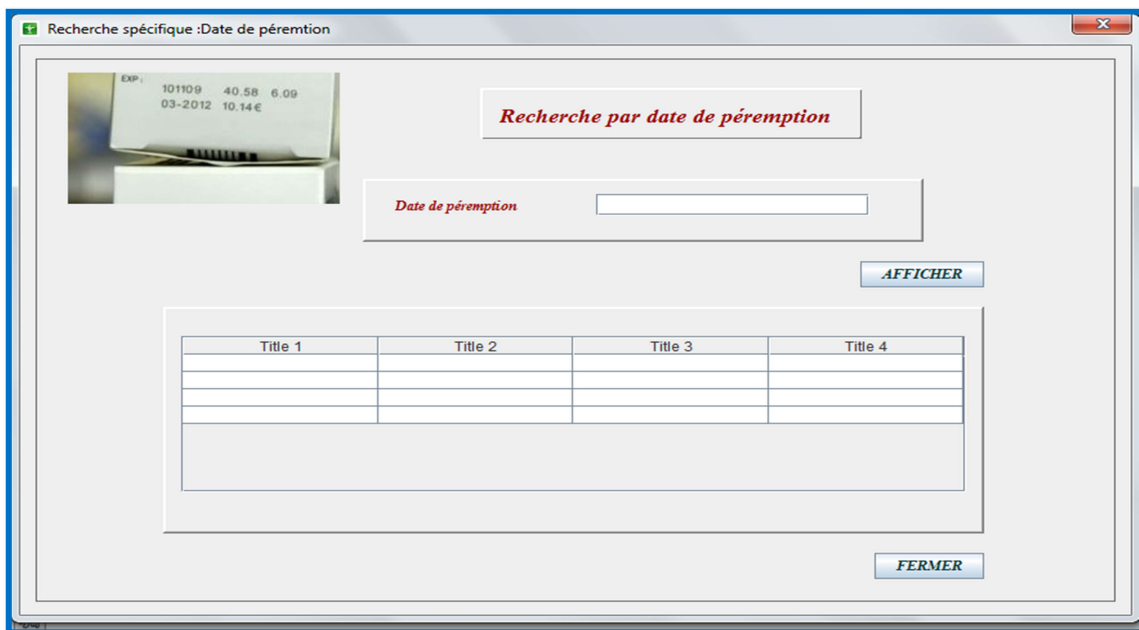


Figure 35 : Interface recherche par date de péremption

III.6.2 Le code source :

Nous suivons la même déclaration et les mêmes méthodes (qui traitent le caractère saisi à partir d'IHM et exécutent HQLQuery et displayResult) et les imports nécessaires de JFrame.

III.6.3 Astuce de plus ! :

Pour plus de sécurité nous allons ajouter une autre interface supplémentaire (mot de passe) pour assurer la sécurité.

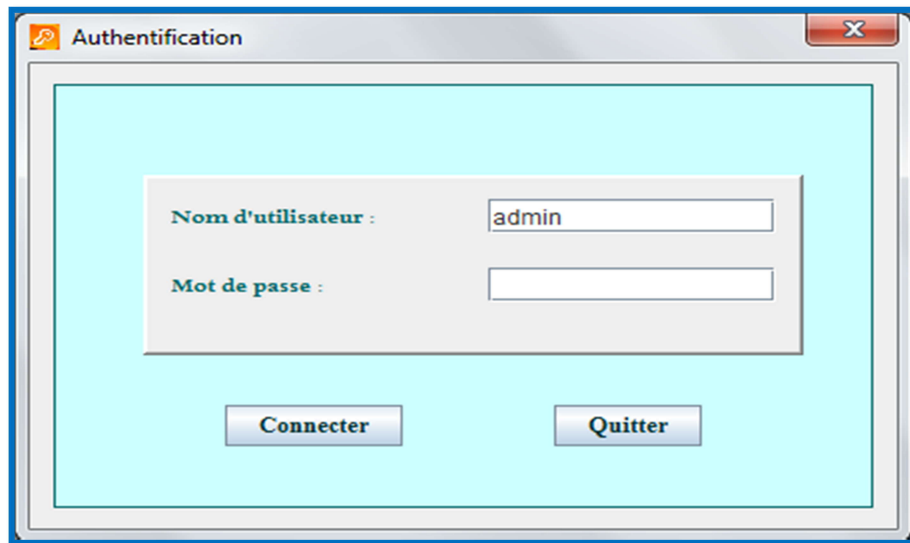
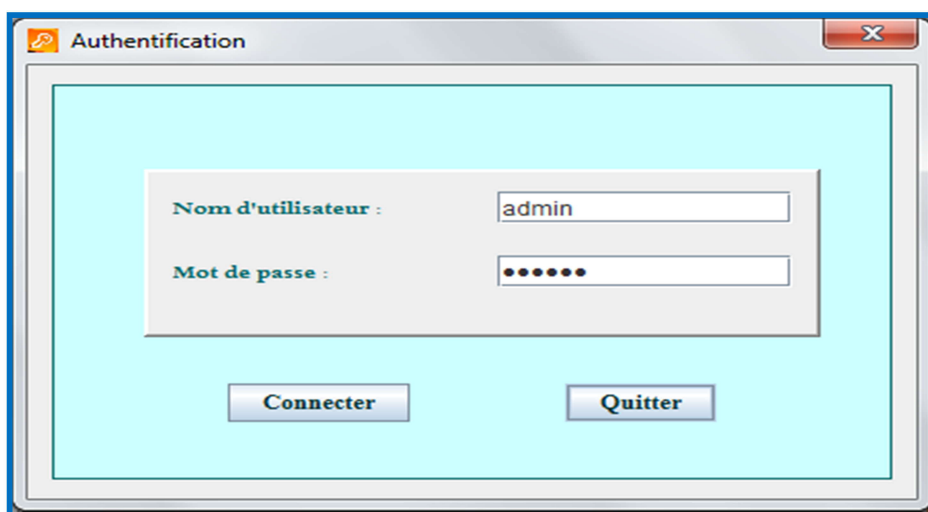


Figure 36 : Interface authentification

III.7 Exécution :

- ✦ Saisir le mot de passe



III.7.1 Premier mode d'exécution : Recherche par désignation

- ✦ Saisir le/les premiers caractères du nom de produit

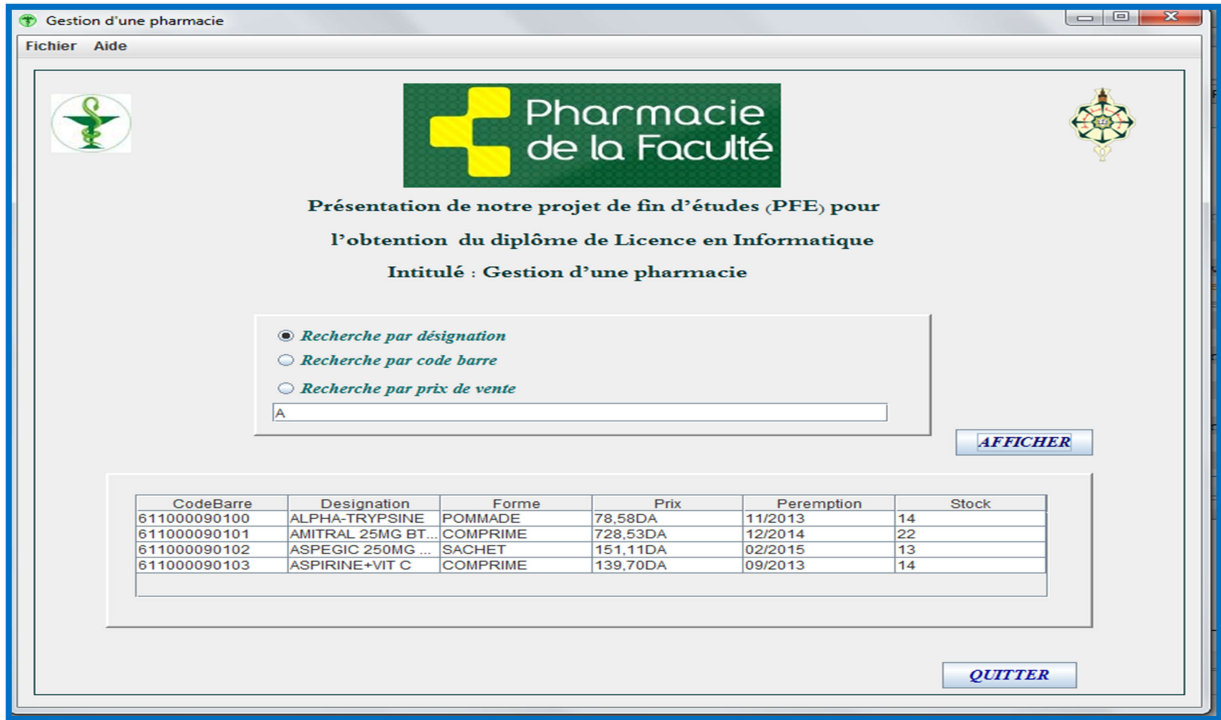


Figure 37 : premier mode d'exécution

III.7.2 deuxième mode d'exécution : Recherche par codeBarre

- ✦ Saisir le code barre du produit

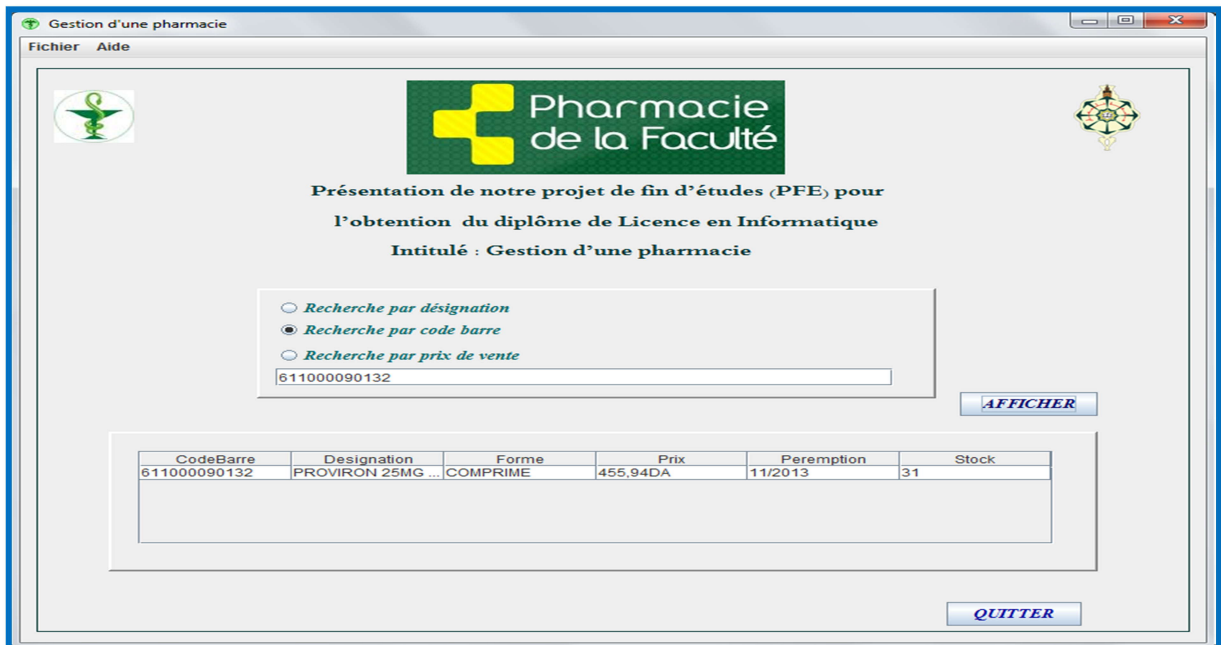


Figure 38 : deuxième mode d'exécution

III.7.3 troisième mode d'exécution : Recherche par prix de vente

- ✦ Saisir le prix de vente du produit

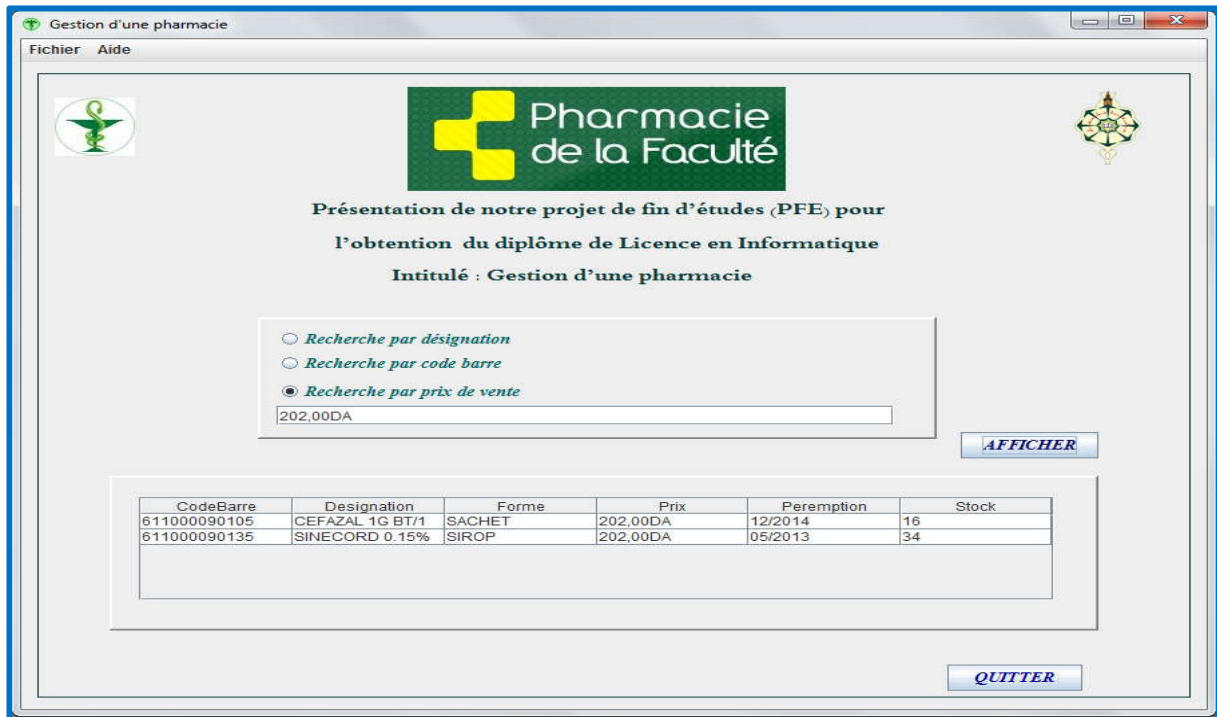
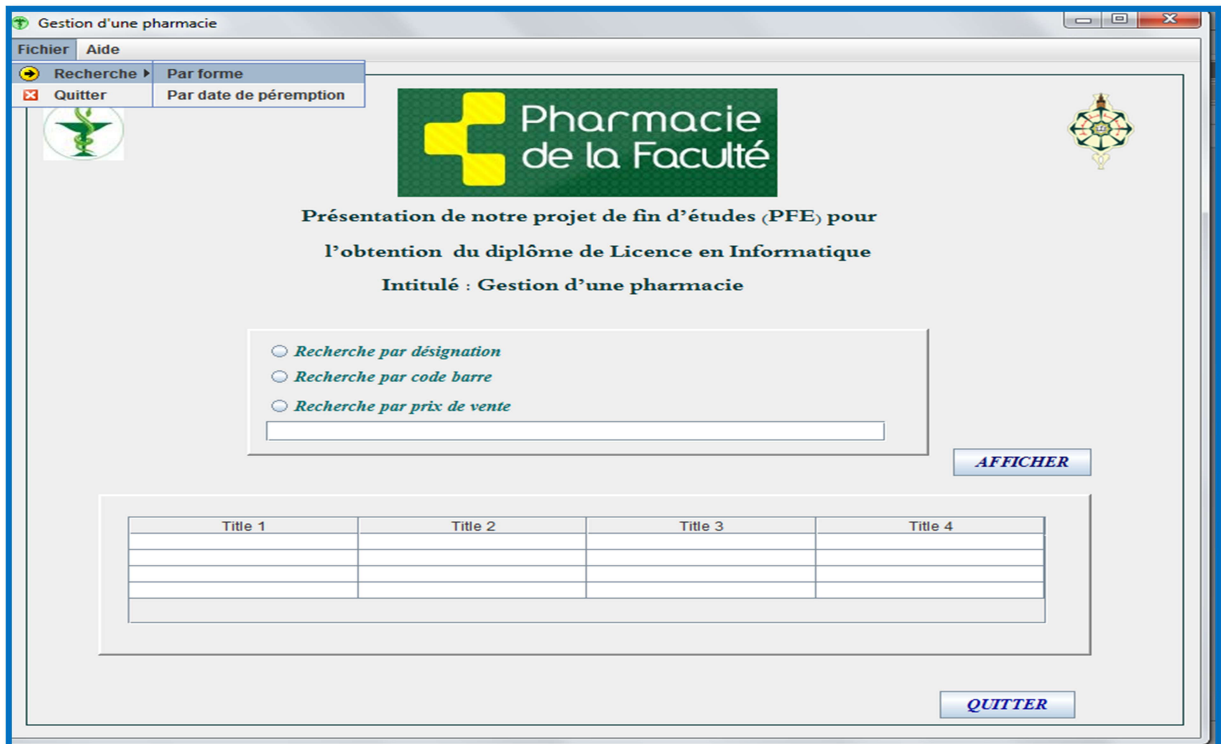


Figure 39 : troisième mode d'exécution

III.7.4 Quatrième mode d'exécution : Recherche par forme



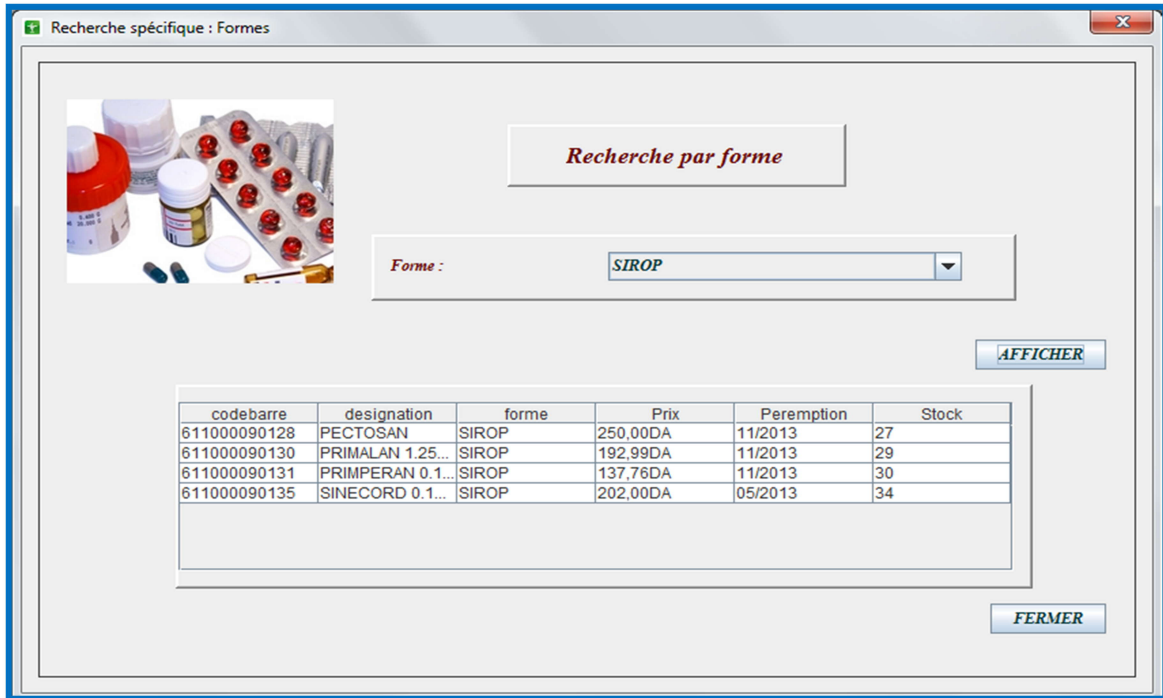
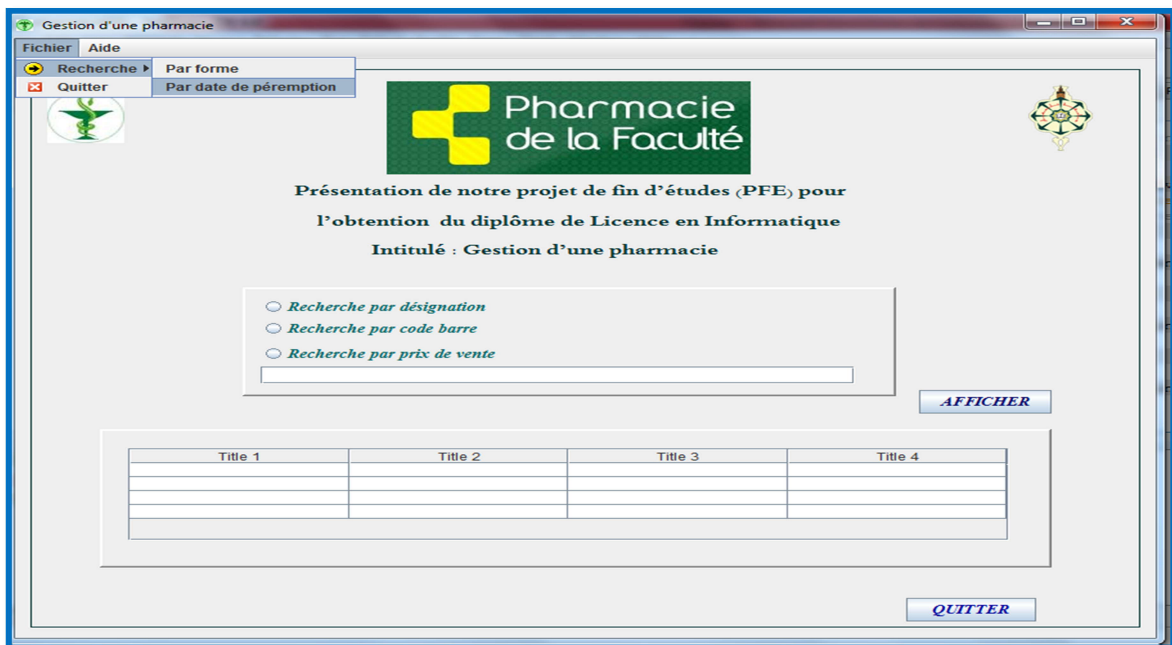


Figure 40: quatrième mode d'exécution

III.7.5 cinquième mode d'exécution : Recherche par date de péremption :

- ✦ Cliquer sur → Fichier → Recherche par → date de péremption



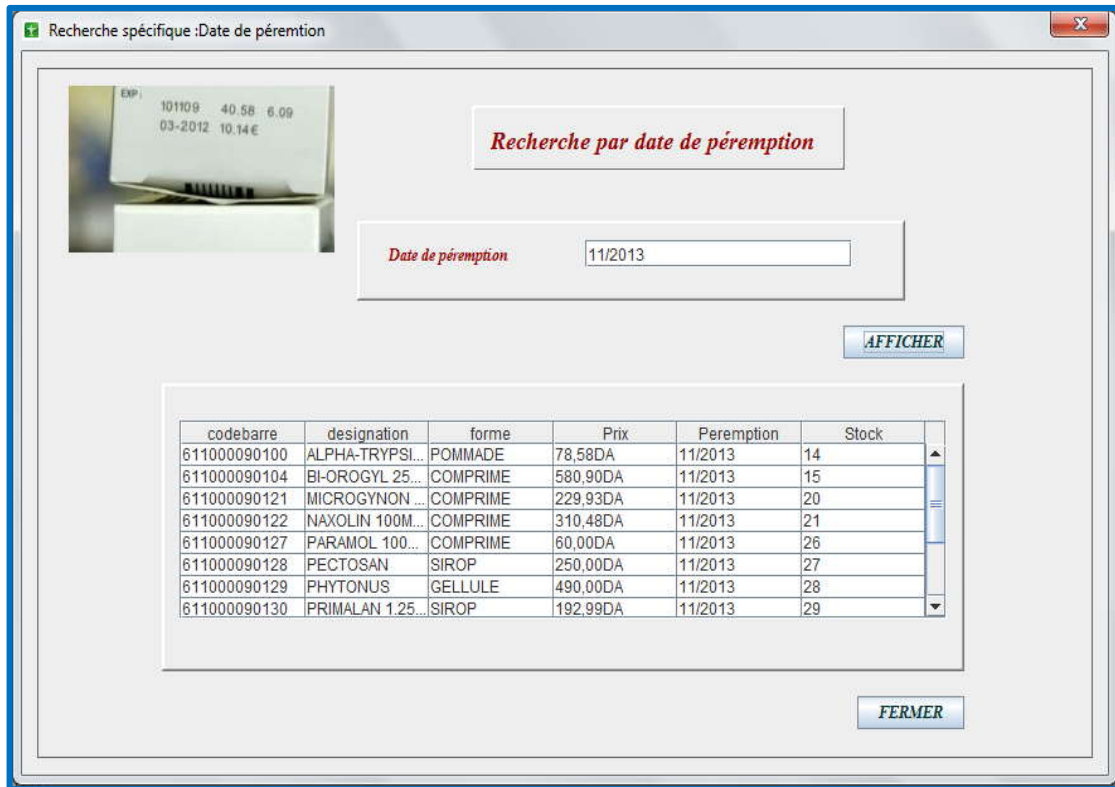
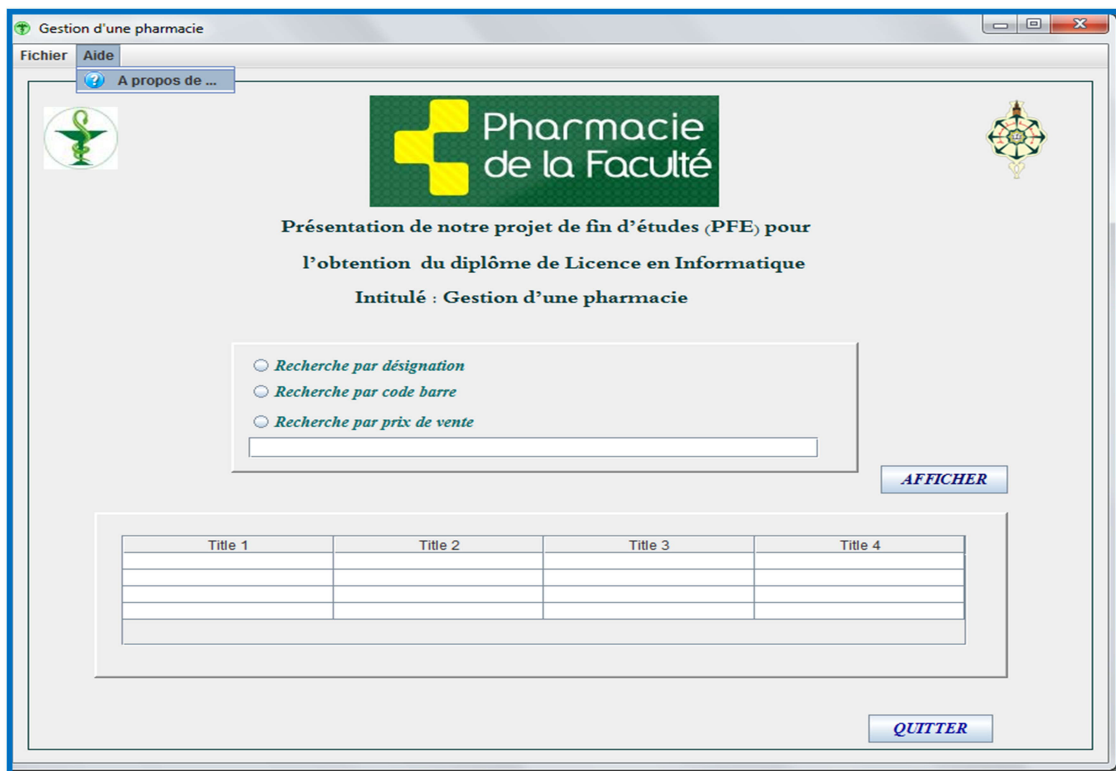


Figure 41: cinquième mode d'exécution

III.7.6 Sixième mode d'exécution :

Cliquer sur → Aide → A propos de



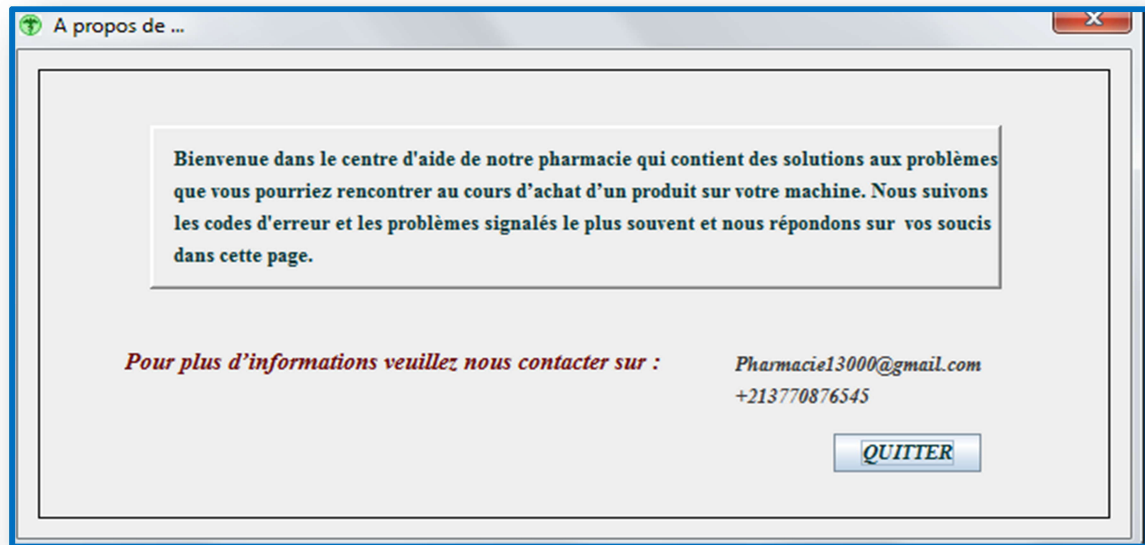


Figure 42: sixième mode d'exécution

III.8 Conclusion :

Dans ce deuxième chapitre, nous avons présenté la description de l'application, la phase de réalisation de notre projet en présentant les étapes de la création de ce projet, la démarche du travail, finalement, nous avons présenté les différents modes d'exécution.

CONCLUSION GENERALE

Conclusion générale

IV. conclusion générale

Dans ce projet de fin d'études, nous avons réalisé une application SWING avec le framework Hibernate, cette application consiste à gérer une vente d'un produit pharmaceutique dans une pharmacie. Nous avons évoqué les outils utilisés pour réaliser notre application ainsi que son déroulement.

Ce PFE nous a aidé à développer nos connaissances et nos idées, nous avons eu le privilège de mieux comprendre le langage de programmation « JAVA » ainsi que le système de base de données et ses fonctionnalités.

Références Bibliographiques

V. Références Bibliographiques

- [1] <http://www.mpl.ird.fr/divha/fr/soft/hyd2002/docs/user/ref/docs/formjava.htm>. Le 14 /04/2013.
- [2] http://www.java.com/fr/download/faq/whatis_java.xml. Le 14 /04/2013.
- [3] http://www.java.com/fr/download/faq/whatis_java.xml. Le 14 /04/2013.
- [4] <http://www.mpl.ird.fr/divha/fr/soft/hyd2002/docs/user/ref/docs/formjava.htm>. Le 14 /04/2013.
- [5] <http://www.mpl.ird.fr/divha/fr/soft/hyd2002/docs/user/ref/docs/formjava.htm>. Le 15 /04/2013.
- [6] http://fr.wikipedia.org/wiki/Java_%28langage%29. Le 15 /04/2013.
- [7] <http://fr.wikipedia.org/wiki/NetBeans>. Le 15 /04/2013.
- [8] <http://fr.wikipedia.org/wiki/NetBeans>. Le 15 /04/2013.
- [9] <http://fr.wikipedia.org/wiki/NetBeans>. Le 16 /04/2013.
- [10] : <http://fr.wikipedia.org/wiki/NetBeans>. Le 16 /04/2013.
- [11] : <http://fr.wikipedia.org/wiki/NetBeans>. Le 16 /04/2013.
- [12] : http://www.microapp.com/definition_easyphp_132.html. Le 17 /04/2013.
- [13] : <http://www.mosaique-info.fr/glossaire-web-referencement-infographie-multimedia-informatique/m-glossaire-informatique-et-multimedia/448-mysql-definition.html>. Le 15 /04/2013.
- [14] : Juliette Dibie, « cours-java-swing pdf » , document PDF, créer le 15/12/2005.
- [15] : https://fr.wikipedia.org/wiki/Abstract_Window_Toolkit. Le 15 /04/2013.
- [16] : http://fr.wikipedia.org/wiki/Swing_%28Java%29. Le 19 /04/2013.
- [17] : <http://www.jmdoudoux.fr/java/dej/chap-frameworks.htm>. Le 19 /04/2013.
- [18] : <http://www-igm.univ-mlv.fr/~dr/XPOSE2004/msamson/>. Le 21 /04/2013.
- [19] : <http://www.commentcamarche.net/contents/222-environnement-client-serveur>.
Le 21 /04/2013.
- [20] : <http://static.commentcamarche.net/www.commentcamarche.net/pictures/cs-images-2-tier.gif>. Le 21 /04/2013.
- [21] : <http://mrproof.blogspot.com/2011/03/larchitecture-client-serveur.html>. Le 21 /04/2013.