CHAPITRE 4 :

MODELISATION ET IMPLEMENTATION

4.1 INTRODUCTION :

Une étape essentielle de tout cycle de développement logiciel ou conceptuel consiste à effectuer une étude préalable. Le but de cette phase est de comprendre le contexte du système. Il s'agit d'éclaircir au mieux les besoins fonctionnels et non fonctionnels, apparaitre les acteurs et identifier les cas d'utilisation. [13]

4.2 SPECIFICATION DES BESOINS

La spécification de besoins constitue la phase de départ de toute application à développer dans laquelle nous allons identifier les besoins de notre application. Nous distinguons des besoins fonctionnels qui présentent les fonctionnalités attendues de notre application et les besoins non fonctionnels pour éviter le développement d'une application non satisfaisante ainsi de trouver un accord commun entre les spécialistes et les utilisateurs pour réussir le projet [13].

4.2.1 Spécification des besoins fonctionnels

Après une étude détaillée de système, cette partie est réservée à la description des exigences fonctionnelles des différents acteurs de l'application. Ces besoins se regroupent dans les diagrammes des cas d'utilisation.

Les besoins utilisateur :

- Authentification de l'utilisateur
- Création et ajout des données pour la génération des fichiers d'entrée (sous forme (.txt,.csv,ect...).
- Génération des règles d'association en utilisant l'un des algorithmes citer dans le chapitre précédant, par utilisation d'un ensemble de paramètres en entrée pour avoir le résultat en sortie.
- Analyse et comparaison en utilisant quelques outils d'analyse des résultats [13].

4.2.2 Spécification des besoins Non fonctionnels

Les besoins non fonctionnels décrivent toutes les contraintes techniques, ergonomiques et esthétiques auxquelles est soumis le système pour sa réalisation et pour son bon fonctionnement. Et ce qui concerne notre application, nous avons dégagé les besoins suivants :

- La disponibilité : l'application doit être disponible pour être utilisé par n'importe quel utilisateur.
- La confidentialité de l'accès à l'application par des utilisateurs bien déterminés.
- La fiabilité : les données fournies par l'application doivent être fiables.
- La convivialité de l'interface graphique : l'application doit fournir une interfaceconviviale et simple pour tout type d'utilisateur car elle présente le premier contact del'utilisateur avec l'application et par le biais de celle-ci on découvrira sesfonctionnalités.
- Une solution ouverte et évoluée : l'application peut être améliorée par l'ajout d'autres modules pour garantir la souplesse, l'évolutivité et l'ouverture de la solution.
- La possibilité de retourner au menu principal de l'application à partir de n'importequelle fenêtre de celle-ci[13].

4.3 METHODOLOGIE ET APPROCHE ADOPTEE

Avant de programmer l'application et se lancer dans l'écriture du code : il faut tout d'abord organiser les idées, les documenter, puis organiser la réalisation en définissant les modules et les étapes de la réalisation. Cette démarche antérieure à l'écriture que l'on appelle modélisation ; son produit et un module.

La modélisation consiste à créer une représentation virtuelle d'une réalité de telle façon à faire ressortir les points auxquels on s'intéresse. Dans le cadre de notre projet on a utilisé la méthodologie UML pour la modélisation des différents diagrammes.

4.3.1 Présentation d'UML :

En regardant les objectifs fixés pour la réalisation du projet, nous remarquons que noussommes face à une application basée sur les algorithmes de génération des résultats et qui devra rester ouverte pour les améliorations futures. De ce fait, il est très important d'utiliser un langage universel pour la modélisation afin de clarifier la conception et de faciliter les échanges. Notre choix est porté sur le langage UML puisqu'il convient pour toutes les méthodes objet et se prête bien à la représentation de l'architecture du système.

UML : UML (Unified Modeling Language) est un langage de modélisation unifié permet de modéliser une application logicielle d'une façon standard dans le cadre de conception orienté objet.

UML permet de couvrir le cycle de vie d'un logiciel depuis la spécification des besoinsjusqu'au codage en offrant plusieurs moyens de description et de modélisation des acteurs etde utilisation système, du comportement des objets, du flot de contrôle internes

auxopérations, des composants d'implémentation et leurs relations, de la structure matérielle et dela distribution des objets et des composants indépendamment des techniquesd'implémentation et peut être mis à jour selon les besoins [13].

4.3.2 Les avantages d'UML

- Universel.
- Adopté par les grandes entreprises.
- Notation unifiée
- Facile à comprendre.
- Adopté par plusieurs processus de développement
- Limite les risques d'erreur.
- N'est pas limité au domaine informatique [13].

4.3.3 LES DIAGRAMMES DE CAS D'UTILISATION

Les diagrammes de cas d'utilisation a pour but de donner une vision globale sur les interfaces de future application. C'est le premier diagramme UML constitué d'un ensemble d'acteurs qui agissent sur des cas d'utilisation et qui décrit sous la forme d'actions et des réactions, le comportement d'un système du point de vue utilisateur.

Acteur : un acteur est un utilisateur qui communique et interagit avec les cas d'utilisation du système. C'est une entité ayant un comportement comme une personne, système ou une entreprise.

Système : cet élément fixe les limites du système en relation avec les acteurs qui l'utilisent (en dehors de système) et les fonctions qu'il doit fournir (à l'intérieur du système).

Cas d'utilisation : un cas d'utilisation représente un ensemble de séquences d'actions àréaliser par le système et produisant un résultat observable intéressant pour un acteurparticulier représenté par des ellipses et limité par un rectangle pour représenter le système.

Dans notre travail, nous avons collectés plusieurs informations et données pour réalisé cette tâche, parmi ces données :

- des fichiers de données contient des articles.
- Le nombre de transactions.

4.3.3.1 Identification des acteurs

Puisque notre système est basé sur une architecture simple, les acteurs qui constituent notre système sont :

Acteur	Rôle
Utilisateur	• Authentifier

	• gestion des transactions (Ajout, Midifier,...) • Générer les fichiers d'entrée sous forme (txt, csv) • Choisir et paramétrer l'algorithme qui doit être exécuter • Analyser et comparer les résultats générer en terme de temps et espace mémoire.

Tableau4.1 Les acteurs de notre système

4.3.3.2 Diagramme de cas d'utilisation globale

Ci-dessous, nous présentons le diagramme de cas d'utilisation pour la compréhension du fonctionnement du système.

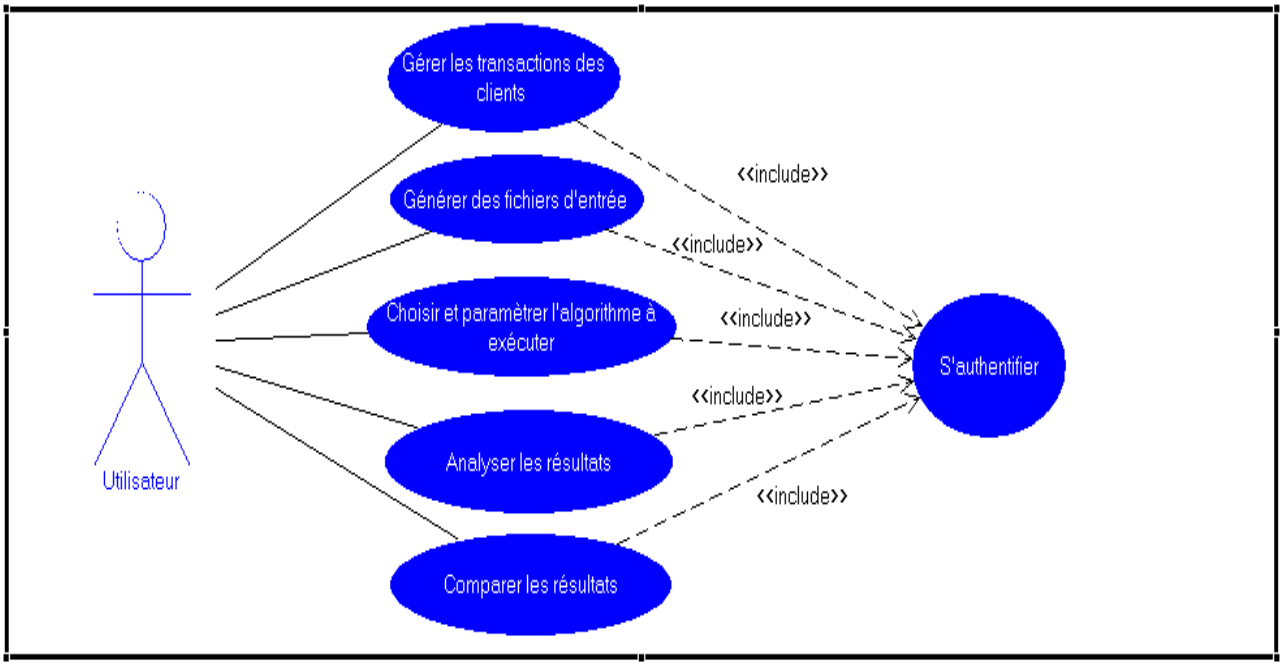


Figure 4.1 Diagramme de cas d'utilisation globale

IV.3.3.2.a Description du cas d'utilisation « **Gérer les transactions** »

Sommaire		
Titre	**Gérer les transactions**	
But	Gérer toutes les opérations nécessaire pour l'ajout de nouveau transaction.	
Acteur	Utilisateur de l'application	
Description des enchainements		
Pré-condition		Post-condition
Utilisateur authentifié		Transaction ajoutés et traités
Scénario nominal		

1- l'utilisateur se connecte au système à travers l'application par login et mot de passe 2- l'utilisateur saisie les champs nécessaires pour l'ajout de nouveau transaction 3- il sauvegarde les données relatif à chaque transaction 4- le système affiche tous les transactions ajoutés par l'utilisateur
Enchaînement alternatif
L'utilisateur n'a pas rempli un champ ou les données sont incorrectes. 1. Le système affiche un message d'erreur. 2. Retour à l'étape 1 du scénario nominal pour lancer à nouveau la connexion.

Tableau4.2:Fiche de description cas d'utilisation : Gérer les transactions des clients

4.3.3.2.b Description du cas d'utilisation « Générer des fichiers d'entrée »

Sommaire	
Titre	**Générer des fichiers d'entrée**
But	Générer les fichiers d'entrée qu'ils seront utilisés comme entrée de l'algorithme à exécuter.
Acteur	Utilisateur de l'application
Description des enchainements	
Pré-condition	Post-condition
Transaction ajoutée	Fichier sous forme de texte est généré
Scénario nominal	
1- l'utilisateur clique sur le bouton pour générer le fichier d'entré. 2- l'utilisateur sauvegarde le fichier dans un chemin qui sera récupéré après lors de l'exécution de l'algorithme 3- Le système affiche un message que le fichier a été généré et sauvegardé avec sucés	
Enchaînement alternatif	
L'utilisateur n'a pas choisi un chemin valide 1. Le système affiche un message d'erreur.	

Tableau4.3 : Fiche de description du cas d'utilisation Générer des fichiers d'entrée

4.3.3.2.c Description du cas d'utilisation « Choisir et paramétrer l'algorithme à exécuter »

Sommaire	
Titre	**Choisir et paramétrer l'algorithme à exécuter**
But	Choisir l'algorithme à exécuter selon des paramètres bien définis comme le support minimale et la confiance minimale avec seuil déterminé préablement.
Acteur	Utilisateur de l'application

Description des enchainements	
Pré-condition	Post-condition
Base de donnée de transaction	Résultat généré
Scénario nominal	
1- l'utilisateur choisit l'algorithme parmi la liste des algorithmes 2- l'utilisateur choisit le fichier généré préalablement comme entrée et définis le chemin de sauvegarde de la sortie (résultat). 3- il saisie les différentes paramètres (support et confiance). 4- il clique sur le bouton d'exécution de l'algorithme, 5- le système affiche l'état d'avancement de l'exécution, ainsi les résultats données.	
Enchaînement alternatif	
L'utilisateur n'a pas rempli un champ ou les données sont incorrectes. 1. Le système affiche un message d'erreur.	

Tableau 4.4: Fiche de description cas d'utilisation Choisir et paramétrer l'algorithme à exécuter

4.3.3.2. d Description du cas d'utilisation « Analyser les résultats» :

Sommaire	
Titre	**Analyser les résultats**
But	Analyser les résultats donnés par différents algorithmes avec des graphes et des tableaux.
Acteur	Utilisateur de l'application
Description des enchainements	
Pré-condition	Post-condition
Résultats sous forme des fichiers texte	Résultat analysée
Scénario nominal	
1- l'utilisateur collecte les différents résultats donnés pour les représenter sous forme des graphes et des tableaux analytiques. 2- le système affiche les différents graphes et tableaux définit par l'utilisateur.	

Tableau4.5 Fiche de description cas d'utilisation Analyser les résultats

4.3.3.2. e Description du cas d'utilisation « Comparer les résultats» :

Sommaire	
Titre	**Comparer les résultats**
But	Comparer les résultats données par les différents algorithmes en terme

	de temps, espace mémoire et nombre de transaction.
Acteur	Utilisateur de l'application
Description des enchainements	
Pré-condition	Post-condition
Résultats sous forme des fichiers texte	Résultat comparé entre les différents algorithmes
Scénario nominal	
1- l'utilisateur compare les résultats données pour les différents algorithmes en fonction de temps, espace mémoire et nombre de transaction. 2- le système affiche les résultats comparé.	

Tableau4.6 : Fiche de description cas d'utilisation Analyser les résultats

4.3.3.2.f Description du cas d'utilisation « S'authentifier » :

Sommaire	
Titre	**S'authentifier**
But	L'authentification a pour but pour sécuriser les accès à l'application .
Acteur	Utilisateur de l'application
Description des enchainements	
Pré-condition	Post-condition
Login et mot de passe	Menu de l'application
Scénario nominal	
1- l'utilisateur se connecte au système à travers l'application par login et mot de passe 2- l'utilisateur saisie les champs nécessaires pour authentifier. 4- le système affiche la menue principale de l'application.	
Enchaînement alternatif	
L'utilisateur n'a pas rempli un champ ou les données sont incorrectes. 1. Le système affiche un message d'erreur.	

Tableau4.7 : Fiche de description cas d'utilisation S'authentifier

Après avoir tracé les grandes lignes de phase de spécification de besoins, mettons l'accent maintenant sur une phase fondamentale dans le cycle de vie d'un logiciel, la phase de conception. Cette phase a pour objectif de déduire la spécification de l'architecture de système.

Cette phase aboutira à la conception et la représentation des diagrammes de séquences et d'activités en se basant sur le langage de modélisation UML.

4.4 LA CONCEPTION GENERALE

4.4.1 Le cycle de développement en v

De nos jours, la méthodologie adoptée dans l'analyse et la conception des systèmes représente un choix stratégique pour les bureau d'études afin de mener à terme les projets tout en respectant les délais annoncés au client et avec la qualité demandée.

Vu l'évolution des besoins des utilisateurs finaux, les applications deviennent de plus en plus complexes et difficiles à concevoir et à développer.

Pour la conception, le développement et la réalisation de notre application, nous avons opté pour l'application du processus de développement V qui demeure actuellement le cycle de vie le plus connu et certainement le plus convenable aux différents type de projets.

Ce processus nous a accompagné du début de projet jusqu'à l'implémentation. Son principe est qu'avec toute décomposition doit être décrite la recomposition, et que toute description d'un composant doit être accompagnée de test qui permettra de s'assurer qu'il correspond à sa description. Le schéma ci-dessous représente les différentes phases du modèle en V :

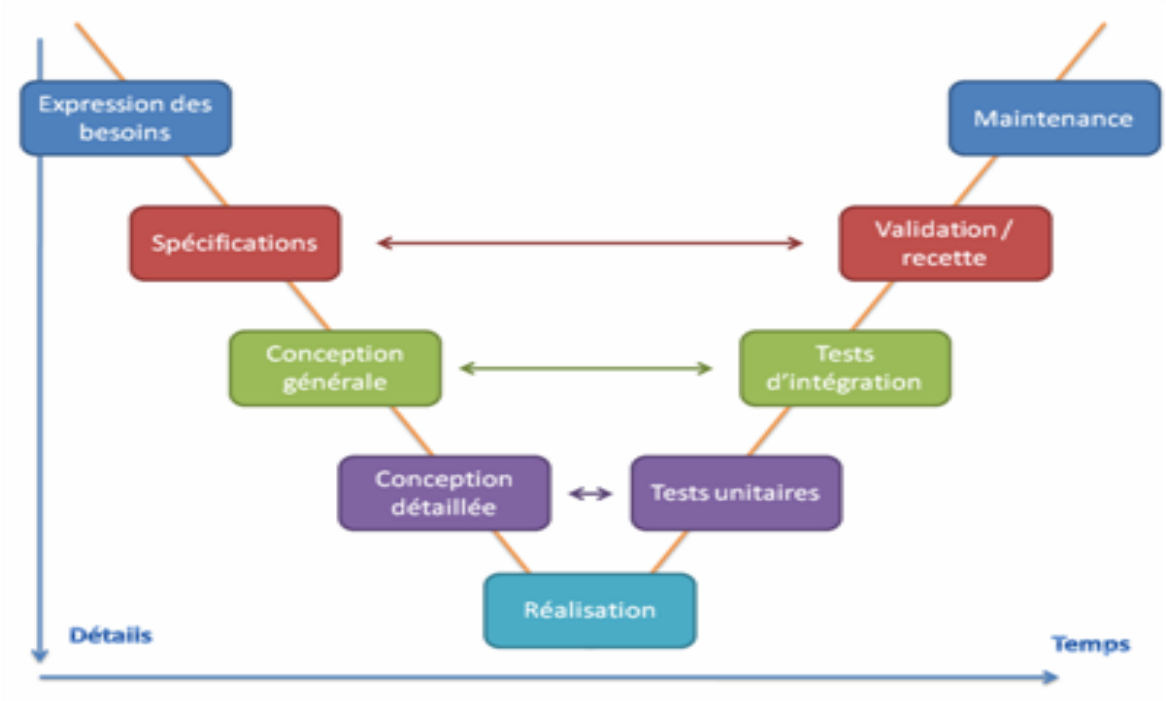


Figure 4.2 Modèle en v

4.5 La ConceptionDétaillée

La conception détaillée met en œuvre itérativement un micro-processus de construction et c'est en cette phase que l'on génère le plus de volume d'informations.

En tant que concepteurs, nous allons élaborer le modèle de conception qui va donner une image « prête à coder » de notre solution.

Cette étape se fera par étape afin d'aboutir à un système fonctionnel reflétant une réalité physique.

4.5.1 Le diagramme de déploiement :

Le diagramme de déploiement définit l'architecture matérielle de l'application. Il présente les périphériques utilisés et la répartition du système sur ces différents éléments. Il montre aussi les liens de communication entre ces diverses entités.

Le diagramme de déploiement de notre application est représenté par le diagramme ci-après :

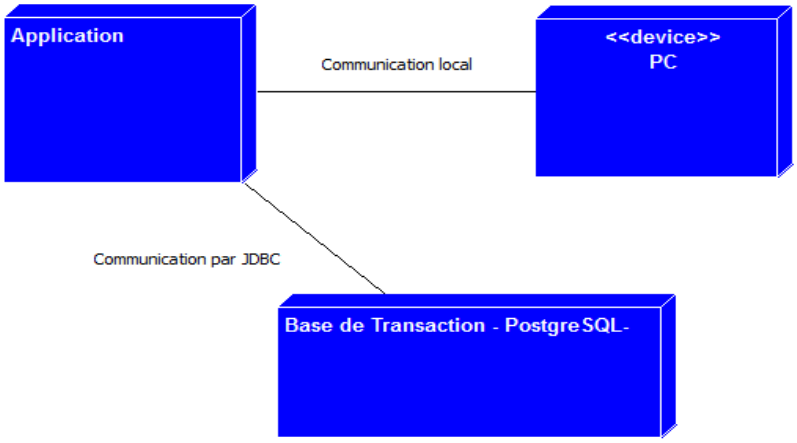


Figure 4.3 Diagramme de déploiement

4.5.2 Les diagrammes de séquence :

Les diagrammes de séquence peuvent servir à illustrer les cas d'utilisations décrits dans la section précédente. Ils permettent de représenter la succession chronologique des opérations réalisées par un acteur et qui font passer d'un objet à un autre pour représenter un scénario.

Dans cette partie, nous allons décrire les scénarios les plus importants ainsi que leursreprésentations par les diagrammes de séquence.

4.5.2.1 le diagramme de séquence « s'authentifier » **:**

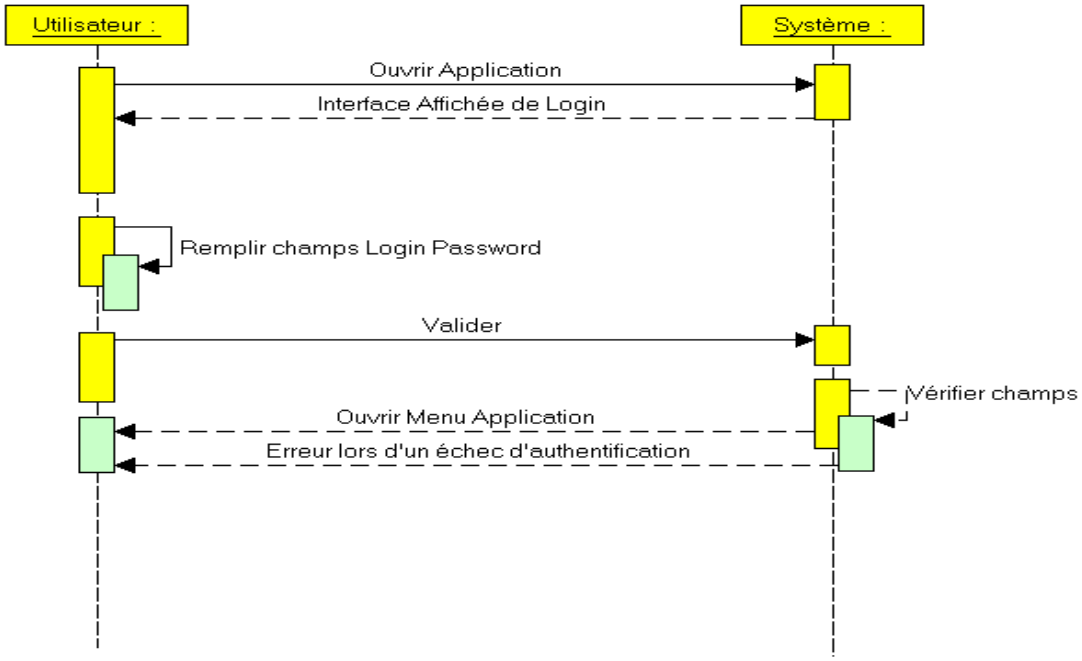


Figure 4.4 Diagramme de séquence « s'authentifier »

4.5.2.2 le diagramme de séquence **«** Gérer les transactions **» :**

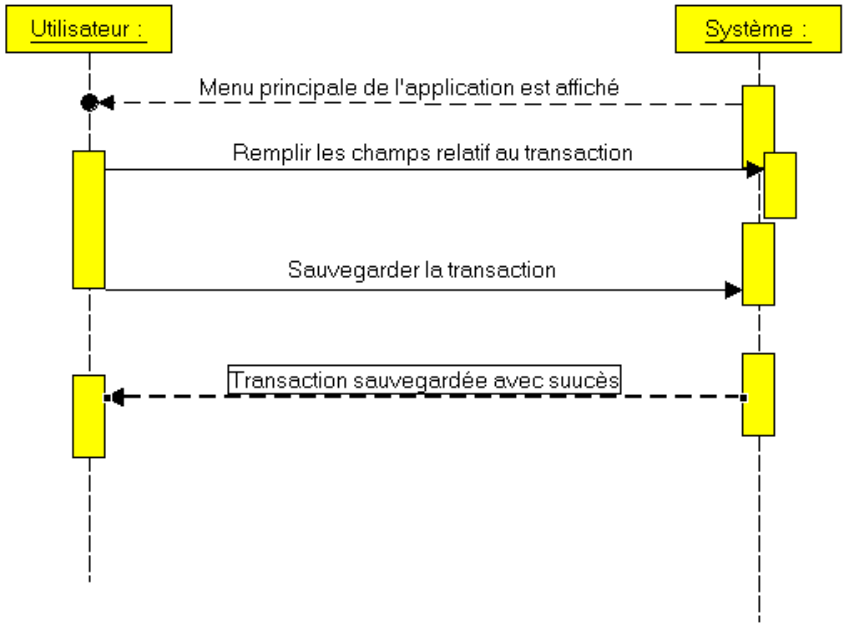


Figure 4.5 Diagramme de séquence «Gérer les transactions »

4.5.2.3 le diagramme de séquence « Générer des fichiers d'entrée » **:**

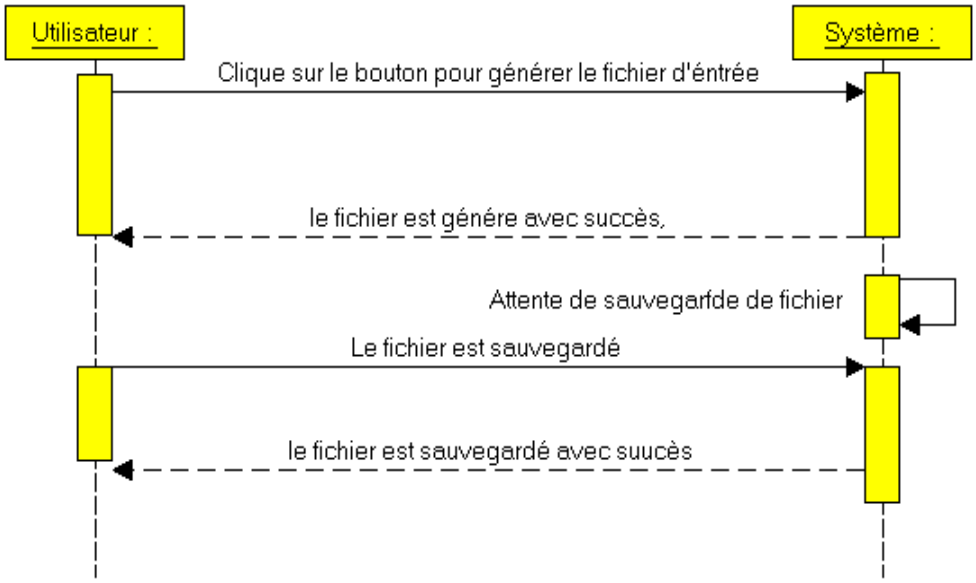


Figure4.6 : Diagramme de séquence «Générer des fichiers d'entrée»

4.5.2.4 le diagramme de séquence«Choisir et paramétrer l'algorithme à exécuter » :

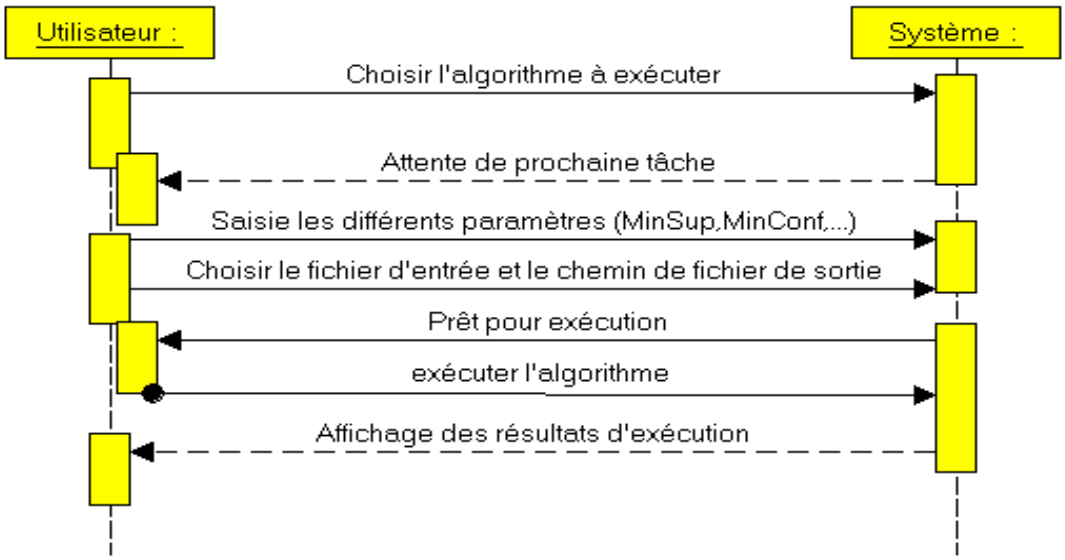


Figure 4.7 : Diagramme de séquence «Choisir et paramétrer l'algorithme à exécuter»

4.5.2.5 le diagramme de séquence «Analyser les résultats » **:**

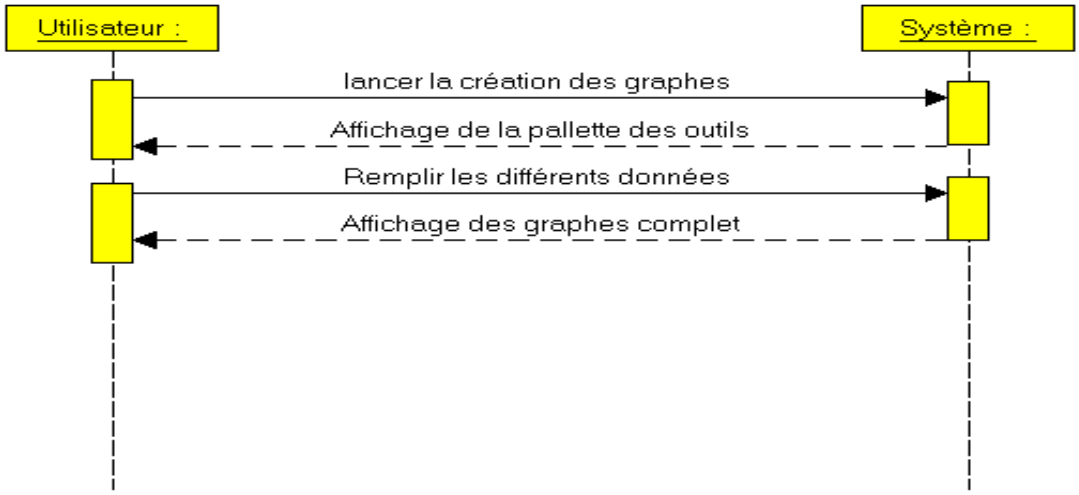


Figure 4.8 : Diagramme de séquence «Analyser les résultats»

4.5.2.6 le diagramme de séquence «Comparer les résultats» **:**

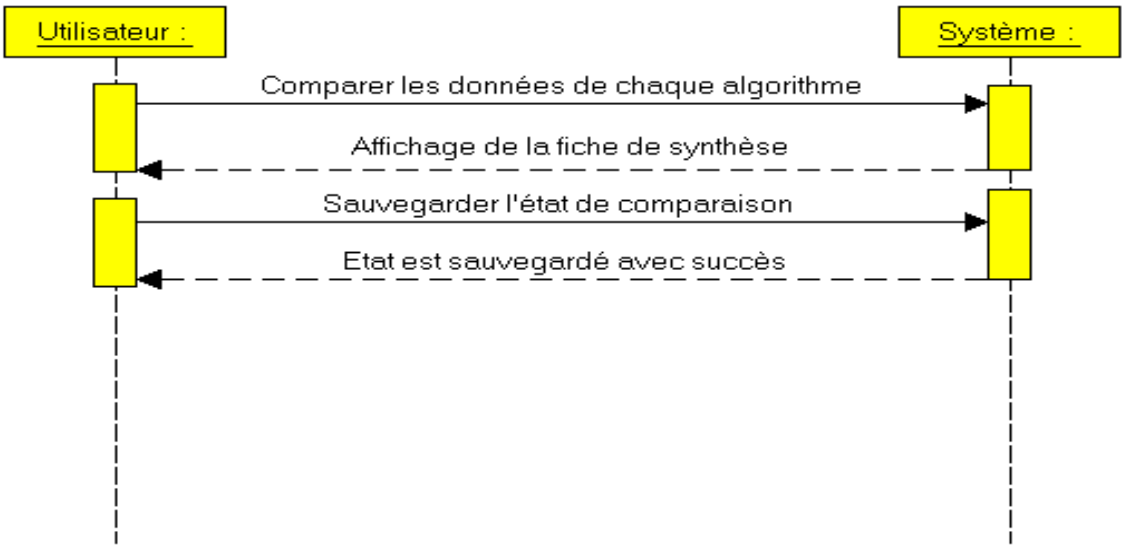


Figure 4.9 : Diagramme de séquence «Comparer les résultats»

4.5.3 Les diagrammes d'activité :

Le diagramme d'activité permet de représenter le déclenchement d'évènements en fonction des états du système et de modéliser des comportements parallèles. Il donne une vision des activités propres à une opération ou à un cas d'utilisation.

Une activité est une opération d'une certaine durée qui peut être interrompue.

Dans ce cas, on va représenter un diagramme d'activité globale qui nous permet de visualiser tous les opérations ainsi tous les tâches nécessaire à la génération des règles d'association ci-après :

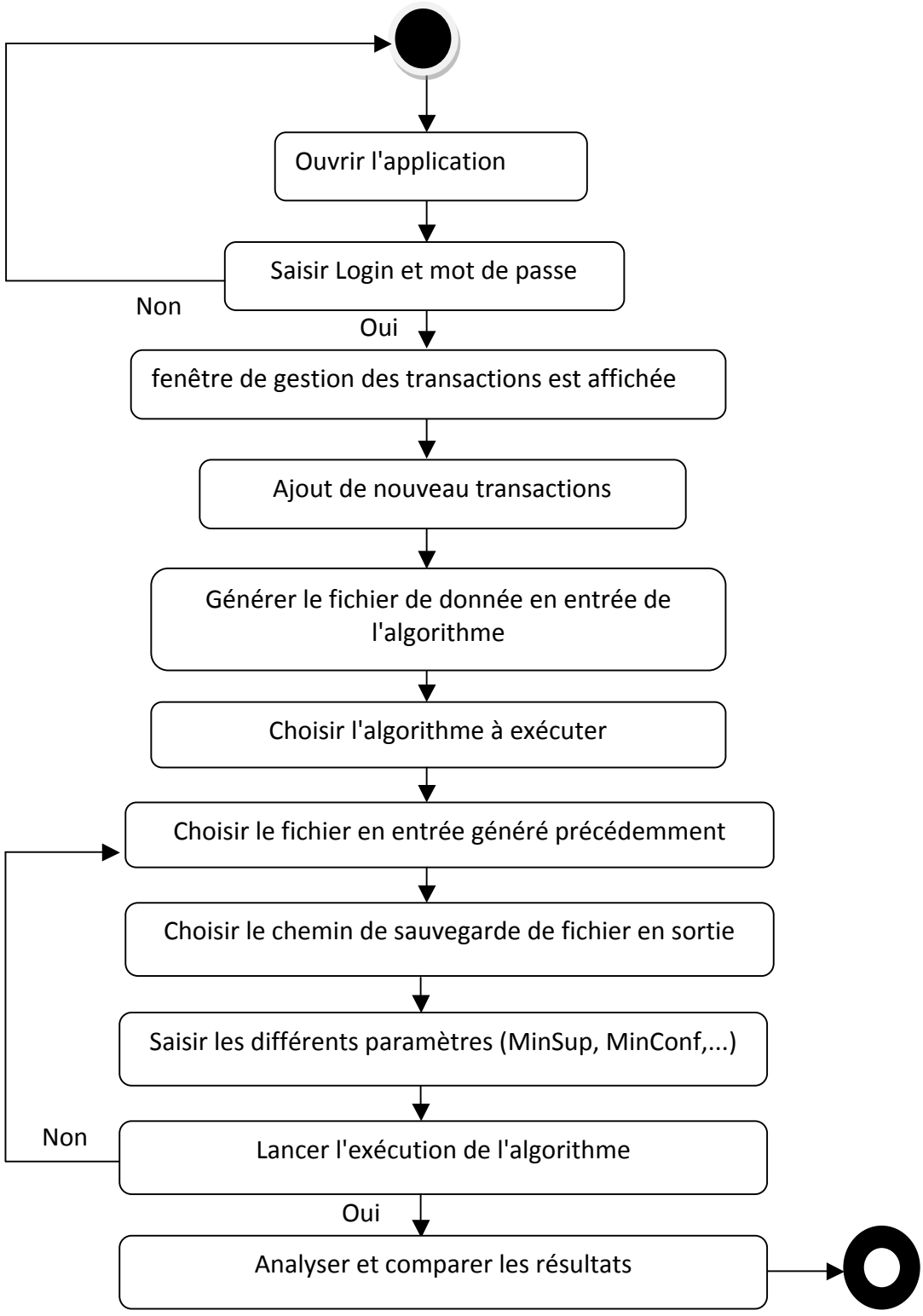


Figure 4.10 : Diagramme d'activité

4.5.4 Diagramme de Classe :

Le diagramme de classes est un schéma utilisé pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci

Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

Les classes peuvent être liées entre elles grâce au mécanisme d'héritage qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre des classes, chacune de ces relations est représentée par un arc spécifique dans le diagramme de classes.

Dans notre cas, on va présenter un diagramme de classe qui nous permet de savoir les différents entités de notre application, ce diagramme est décrit ci-après :

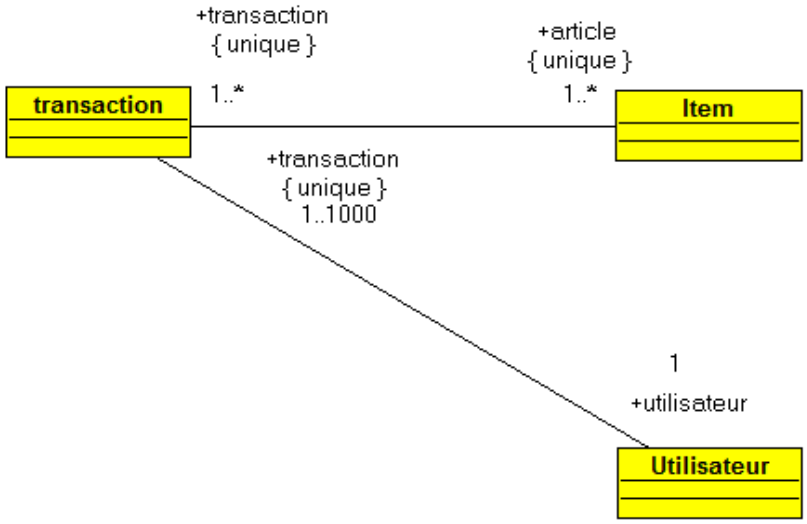


Figure 4.11 Diagramme de classe

4.6 Implémentation :

Cette partie constitue le dernier volet de ce projet,après avoir terminé la phase de spécification et conception, la solution étant déjà choisie et étudiée, il nous reste que de se décider dans quel environnement nous allons travailler, exposer les choix techniques utilisés et le langage adopté, et présenter l'implémentation et les tests réalisés ainsi les résultats obtenus.

4.6.1 Choix technique :

4.6.1.1 Choix du langage de programmation :

Java

Java est un langage de programmation orienté objet, développé par Sun Microsystems et destiné à fonctionner dans une machine virtuelle, il permet de

créer des logiciels compatibles avec des nombreux systèmes d'exploitation.

Java et non seulement un langage de programmation puissant conçu pour être sûr, inter plateformes et international, mais aussi un environnement de développement qui est continuellement étendu pour fournir des nouvelles caractéristiques et des bibliothèques permettant de gérer de manière élégante des problèmes traditionnellement complexes dans les langages de programmation classiques, tels que le multithreading, les accès aux bases des données, la programmation réseau, l'informatique répartie.

En plus java est considéré comme un langage adaptable aux plusieurs domaines puisque une application web implémentée par celle-ci peut avoir des extensions ou des modifications dans le futur. De plus, java permet de réduire le temps de développement d'une application grâce à la réutilisation du code développé.

la version que nous avons utilisés est **JDK 7** (java developpement kit).

4.6.1.2 Choix de l'architecture de l'application**:**

L'architecture à 1-tiers consiste à mettre tous les composants nécessaires à une application logicielle ou une technologie sur un seul poste ou plate-forme. Ce type d'architecture est souvent opposé à l'architecture multi-niveaux ou l'architecture à trois tiers qui est utilisé pour certaines applications Web et d'autres technologies où différentes couches de présentation, ou d'accès aux données sont logés séparément.

Fondamentalement, une architecture 1-tiers conserve tous les éléments d'une application, y compris l'interface, exécutable et les données, en un seul endroit. Les développeurs voient ces types de systèmes comme le plus simple et le plus direct. Certains experts décrivent comme des applications qui pourraient être installées et exécutées sur un seul ordinateur.

La nécessité pour les modèles distribués pour les applications Web et des solutions d'hébergement Cloud a créé de nombreuses situations où les architectures à un seul niveau ne sont pas suffisantes. Cela fait à trois niveaux ou de l'architecture multi-niveaux pour devenir plus populaire. Les avantages d'une solution multi-niveaux sont souvent évidents. Ils peuvent fournir une meilleure sécurité, une meilleure performance et plus d'évolutivité. Toutefois, l'appel d'une architecture 1-tiers peut se rapporter aux coûts qui sont impliqués, où il serait plus logique de garder des applications plus simples contenues dans une plate-forme facile.

L'architecture 1-tiers peut être bénéfique lorsque nous traitons avec des données qui se rapportent à un seul utilisateur (ou un petit nombre d'utilisateurs) et nous avons une quantité relativement faible de données. Ils sont peu coûteux à déployer et à maintenir [12].

A cet effet, notre application sera utilisée sur un seul poste ou un seul plateforme, et on va opter pour cette architecture pour les raisons suivants :

- Notre application est basé sur un petit nombre d'utilisateurs
- Notre cas n'est pas coûteux en terme de matériels et sécurité
- On peut gérer notre application avec un seul poste performant en terme de mémoire et vitesse.

4.6.2 ENVIRONNEMENT LOGISTIQUE :

4.6.2.1 Environnement de développement :

Java EE : nous avons préparé une plateforme java ee, qui est la proposition de SUN pour le développement et la mise en œuvre d'applications d'entreprise multi niveaux. Elle permet de simplifier le développement d'applications en environnement client léger et permet aussi au programmeur le développement normalisé de composants modulaires réutilisables.

Cette plate-forme Java EE est une suite robuste de services middleware qui simplifie la tâche des développeurs d'applications. Elle repose sur les technologies existantes de la plate-forme Java SE afin de faciliter la création d'applications réparties.

Les principales caractéristiques de Java EE sont :

· C'est une spécification pour le langage de programmation Java plus particulièrement destinée aux applications d'entreprise.

· Une grande variété de plateformes et de systèmes d'exploitation sont capables de faire fonctionner une application Java EE.

· Il existe un grand nombre d'IDE différents pour le développement ;

· Toutes les spécifications sont publiques. [13]

Outils de développement IDE ECLIPSE :

Eclipse est un environnement de développement intégré libre extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.

La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plugin : toutes les fonctionnalités de cet atelier logiciel sont développées entant que plug-in.

Dans le cadre de notre projet, nous avons utilisé la version Eclipse Java EE IDE Mars.1 Release (4.5.1), avec plusieurs plugins.Parmi ces plugin, on cite les suivants :

- **Windows Builder** est un composant de Design qui appartiens à la bibliothèque SWT et Swing, il est très facile de créer des applications Java GUI sans dépenser beaucoup de temps à écrire du code, il permetd'Utiliser le concepteur qui possède des outils visuels pour créer des formes simples, des tableaux; le code Java sera généré pour vous, Ajouter facilement des contrôles par glisser-déposer, ajouter des gestionnaires d'événements pour vos commandes, modifier diverses propriétés des contrôles en utilisant un éditeur de propriété.

  Windows Builder est construit comme un plug-in pour Eclipse et les divers IDEs basés sur Eclipse (RAD, RSA, MyEclipse, JBuilder, etc.). Le plug-in construit un arbre de syntaxe abstraite (AST) pour naviguer dans le code source et utilise FEM pour afficher et gérer la présentation visuelle.
- Le Package Data Mining Java (**JDMP**) est une bibliothèque d'une source ouverte pour l'analyse des données et l'apprentissage de la machine. Il facilite l'accès à des sources de données et des algorithmes d'apprentissage machine (par exemple le regroupement, la régression, la classification, les modèles graphiques, optimisation, règles d'association) et fournit des modules de visualisation. **JDMP** fournit un certain nombre d'algorithmes et d'outils, mais également des interfaces vers d'autres paquets apprentissage machine et extraction des données (Weka, LibLinear, ElasticSearch, libsvm, Mallet, Lucene, Octave).
- Package **JfreeChart** de JAVA**[16]:**JFreeChart est une bibliothèque graphique 100% gratuit qui le rend facile pour les développeurs d'afficher des graphiques de qualité professionnelle dans leurs applications. il possède des fonctionnalités vaste comprend :
  - Une API cohérente et bien documenté, soutenant un large éventail de types de graphiques;
  - Une conception flexible qui est facile à étendre, ainsiles applications destinées à l'architecture client/serveur
  - Un support pour de nombreux types de sortie, y compris les composants Swing et JavaFX, des fichiers d'image (y compris PNG et JPEG), et les formats de fichiers graphiques vectoriels (y compris PDF, EPS et SVG);
  - JFreeChart est open source ou, plus précisément, le logiciel libre. Il est distribué sous les termes de la licence publique GNU Lesser General public licence (LGPL), qui permet une utilisation dans des applications propriétaires.

4.6.2.2 système de gestion de base de données :



PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD.

Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB, MySQL et Firebird), ou propriétaires (comme Oracle, Sybase, DB2, Informix et Microsoft SQL Server). Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises.

Ce SGBDRO utilise des types de données modernes, dit composés ou enrichis suivant les terminologies utilisées dans le vocable informatique usuel. Ceci signifie que PostgreSQL peut stocker plus de types de données que les types simples traditionnels entiers, caractères, etc. L'utilisateur peut créer des types, des fonctions, utiliser l'héritage de type, etc.

PostgreSQL est plus avancé que ses concurrents dans la conformité aux standards SQL (énoncé vrai entre 2012 et 2015). PostgreSQL est pratiquement conforme (de plus en plus conforme) aux normes ANSISQL 89, SQL 92 (SQL 2), SQL 99 (SQL 3), SQL:2003 et SQL:2008[5]. Il fonctionne sur diverses plates-formes matérielles et sous différents systèmes d'exploitation [14].

PostgreSQL fonctionne sur Solaris, SunOS, Mac OS X, HP-UX, AIX, Linux, IRIX, Digital Unix, BSD, NetBSD, FreeBSD, OpenBSD, SCO unix, NeXTSTEP, UnixWare et toutes sortes d'Unix. Depuis la version 8.0, PostgreSQL fonctionne également nativement sur Windows. Avant la version 8, il fallait une couche de compatibilité POSIX (par exemple cygwin) pour faire fonctionner PostgreSQL sur ce système d'exploitation.

PostgreSQL est largement reconnu pour son comportement stable, proche de *Oracle*. Mais aussi pour ses possibilités de programmation étendues, directement dans le moteur de la base de données, via PL/pgSQL. Le traitement interne des données peut aussi être couplé à d'autres modules externes compilés dans d'autres langages.

4.6.2.3 Les données et structure de données :

Les données utilisés pour exécuter nos algorithmes sont sous format (*.txt, .csv*), ces données sont importés depuis des base de données réels qui a été alimentés par les utilisateurs, ces fichiers contiennent un nombre de transactions entre 3.000 à 68.000 transactions, ils contiennent des articles (items) numérotés, chaque item correspond à une article bien défini, cet base de transaction contient des transactions de taille différente.

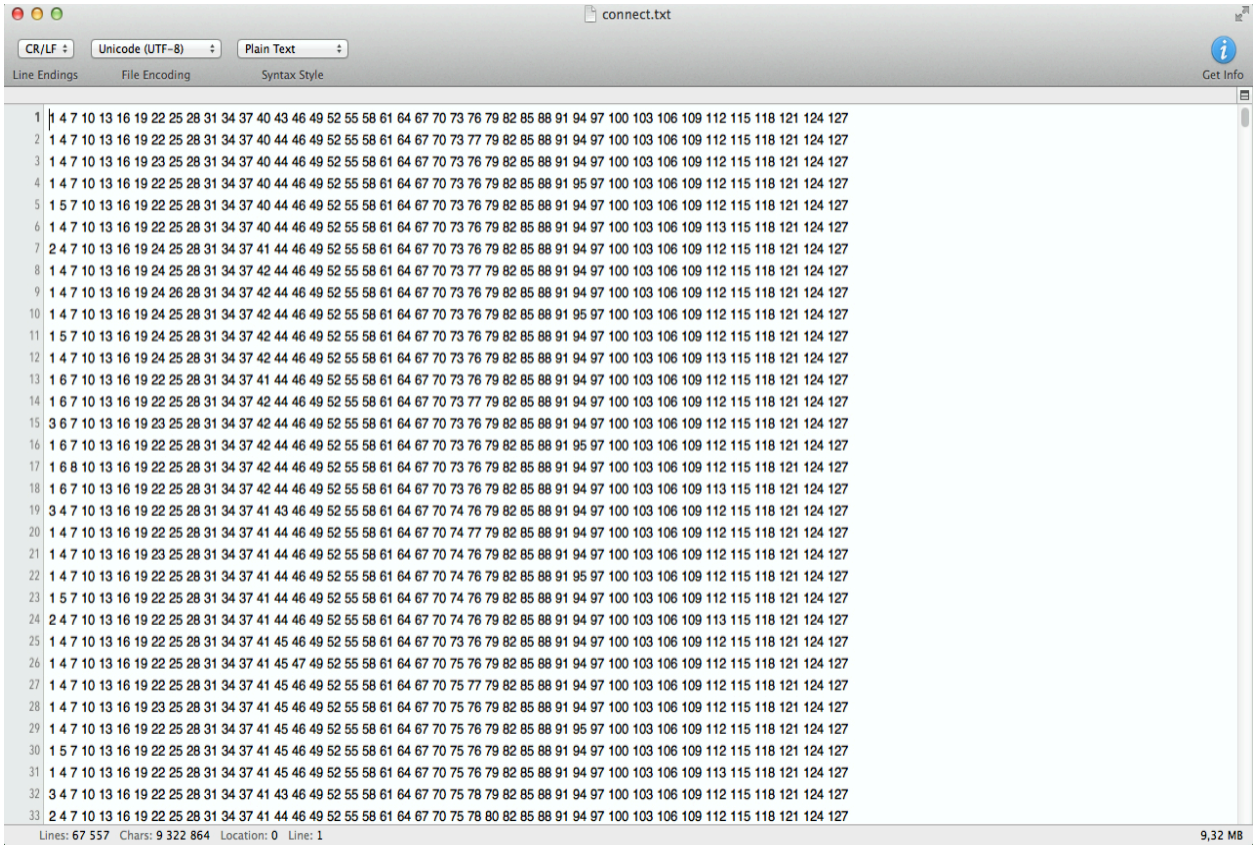


Figure 4.12 Fichier de base de transaction au format texte

Structure de donnée:

Ces algorithmes ont été implémentéespar une structure de donnée s'appelle Red Black tree, cette dernière est unearbre de recherche binaire avec un bit supplémentaire de stockage par nœud: sa couleur, qui peut être soit rouge ou noir. En limitant la façon dont les nœuds peuvent être de couleur sur tout chemin à partir de la racine à une feuille, ces arbres assurent qu'un tel chemin est plus de deux fois plus long que tout autre, de sorte que l'arbre est à peu près équilibré.

Chaque nœud de l'arbre contient la couleur des champs, la clé, partie gauche, partie droite, et p. Si un enfant ou le parent d'enfant n'existe pas, le champ de pointeur correspondant du nœud contient la valeur Null [17].

Une arbre de recherche binaire est un arbre de couleur rouge-noir si elle satisfait les propriétés suivantes:

- Chaque nœud prend la couleur rouge ou noir.
- La racine est noire.
- Chaque feuille prend la valeur (NIL) est noir.
- Si un nœud est rouge, alors ses deux enfants sont noirs.

- Pour chaque nœud, tous les chemins du nœud vers les feuilles contiennent le même nombrede nœuds noirs.

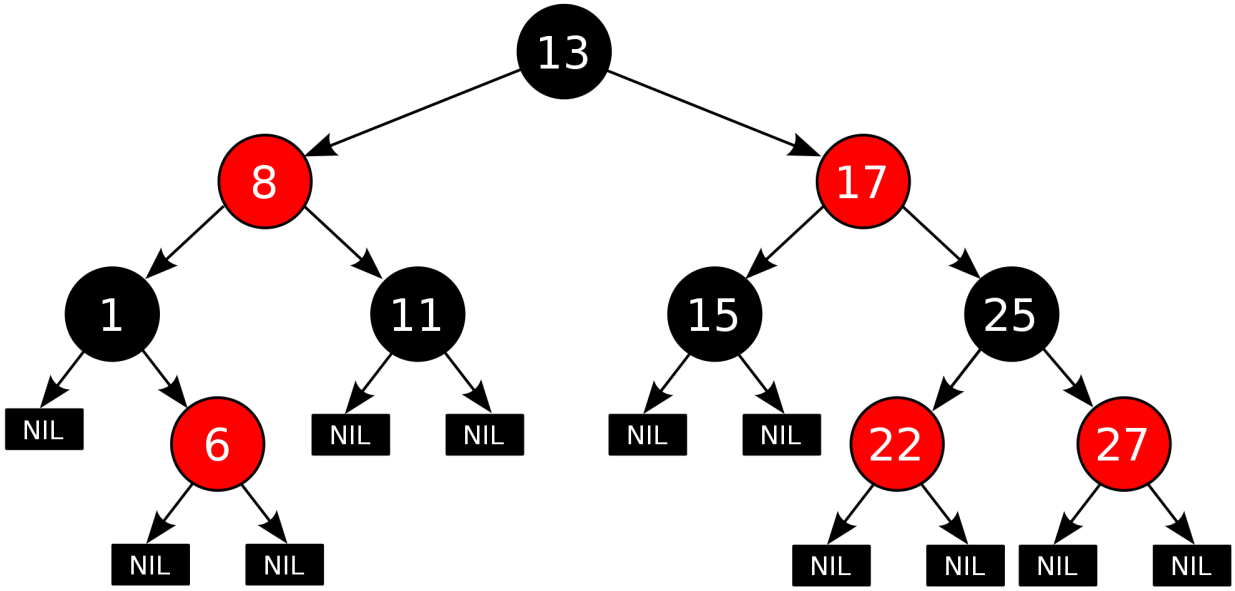


Figure4.13 Arbre coloré avec des nœuds rouges et noirs

Cet arbre possède plusieurs opérations, comme l'insertion, la suppression et recherche, on va s'intéresser par l'opération d'insertion, cet opération possède plusieurs cas à traiter [17]:

- N est le nœud racine, c.à.d. le premier nœud est en rouge-noir
- Le parent de N est noir
- Le parent (P) de N et son oncle (U) sont rouges
- N est ajouté à droite de l'enfant gauche de grand-parent, ou N est ajouté à gauche de l'enfant droit de grand-parent (P est rouge et U est noir)
- N est ajouté à gauche de l'enfant gauche de grand-parent, ou N est ajouté à droite de l'enfant droit de grand-parent (P est rouge et U est noir)

Le pseudo code de cette opération est le suivant :

Procédure Insertion élément dans l'arbre Rouge-Noire

Procedure Insert(T, z)

```
1. y ← nil[T]
2. x ← root[T]
3. tantque  x ≠ nil[T] faire
4. y ← x
5. Si key[z] < key[x] Alors
6. x ← left[x]
7.Sinon  x ← right[x]
8. p[z] ← y
9.Si y = nil[T] Alors
10. root[T] ← z
```

```
11.                 SinonSI key[z] < key[y] Alors
12. left[y] ← z
13.                         Sinon right[y] ← z
14. left[z] ← nil[T]
15. right[z] ← nil[T]
16. color[z] ← RED
17. Procedure_Insert_FIXUP(T, z)
```

Procedure_Insert_FIXUP (T, z)

```
1 Tantque  color[p[z]] = RED faire
2 SI p[z] = left[p[p[z]]] Alors
3            y ← right[p[p[z]]]
4           SI color[y] = RED Alors
5                   color[p[z]] ← BLACK
6                   color[y] ← BLACK
7                   color[p[p[z]]] ← RED
8                  z ← p[p[z]]
9          Sinon Si z = right[p[z]] Alors
10                   z ← p[z]
11                   LEFT-ROTATE(T, z)
12                   color[p[z]] ← BLACK
13                   color[p[p[z]]] ← RED
14                   RIGHT-ROTATE(T, p[p[z]])
15                 Sinon
16            color[root[T]] ← BLACK
```

LEFT-ROTATE(T, x)

```
1 y ← right[x]
2 right[x] ← left[y]
3 p[left[y]] ← x
4 p[y] ← p[x]
5          Si p[x] = nil[T] Alors
6                   root[T] ← y
7          Sinon SI x = left[p[x]] Alors
8 left[p[x]] ← y
9                  Sinon right[p[x]] ← y
10 left[y] ← x
11 p[x] ← y
```

RIGHT-ROTATE(T, x)

```
1 y ← left[x]
2 left[x] ← right[y]
3 p[right[y]] ← x
4 p[y] ← p[x]
5          Si p[x] = nil[T] Alors
6                   root[T] ← y
7          Sinon SI x = right[p[x]] Alors
8                         right[p[x]] ← y
9                  Sinon left[p[x]] ← y
10                         right[y] ← x
11                       p[x] ← y
```

Maintenant, on va écrie le code java correspond à ce pseudo code :

La procédure Insertion en JAVA

```
private void leftRotate(Node x) {
        Node y = x.right;
        x.right = y.left;
        if(!y.left.equals(NULL)){
            y.left.parent = x;
        }
        y.parent = x.parent;
        if(x.parent.equals(NULL)){
            root = y;
        }else if(x.equals(x.parent.left)){
            x.parent.left = y;
        }else{
            x.parent.right = y;
        }
        y.left = x;
        x.parent = y;
    }
        private void rightRotate(Node x) {
        Node y = x.left;
        x.left = y.right;
        if(!y.right.equals(NULL)){
            y.right.parent = x;
        }
        y.parent = x.parent;
        if(x.parent.equals(NULL)){
            root = y;
        }else if(x.equals(x.parent.right)){
            x.parent.right = y;
        }else{
            x.parent.left = y;
        }
        y.right = x;
        x.parent = y;
    }
public void add(T element) {
        // create element
        Node z = new Node();
        z.key = element;
        Node y = NULL;
        Node x = root;
        while (x != NULL) {
            y = x;
            int compare = z.key.compareTo(x.key);
            if (compare < 0) {
                x = x.left;
            } else {
                if (compare == 0 && !allowSameElementMultipleTimes) {
                    return;
```

```
            }
            x = x.right;
        }
    }
    z.parent = y;
    if (y == NULL) { // case of an empty tree
        root = z;
    } else if (z.key.compareTo(y.key) < 0) {
        y.left = z;
    } else {
        y.right = z;
    }
    size++;
    // code specific to red black trees
    z.left = NULL;
    z.right = NULL;
    z.color = RED;
    insertFixup(z);
}
```

4.7 Travaille réalisé :

4.7.1Description de l'application :

La conception des interfaces notre applicationest une étape très importante puisque toutes les interactions avec la couleur de l'application passent à travers ces interfaces, nous devons alors guider l'utilisateur avec les messages d'erreur et avec des notifications si besoin.

Dans cette partie, nous allons présenter quelques cas d'utilisations, sous forme d'un guide utilisateur.

Pour accéder à notre application, on doit s'authentifier. Comme toute application, la sécurité d'accès est nécessaire. La figure ci-après donne l'interface à travers laquelle l'utilisateur s'identifie. Il saisit son nom dutilisateur et son mot de passse, si les informations saisis correspond aux informations enregistrés dans la base de donnée une interface principale s'affiche.



Figure 4.14 : Interface d'authentification

L'interface pricipale permet d'accéder à l'interface suivantequi contient les différents algorithmes à exécuter en cliquant sur le button Commencer, cette interface est représentée ci-après :



Figure 4.15 : Interface principale

L'interface suivante est l'interface qui regroupe les algorithmes qui doit être exécuter par l'utilisateur, en passant plusieurs paramètres, parmi ces paramètres en cite :

- Le fichier d'entrée : c'est un fichier de plusieurs transactions qui contient des items en extension .txt.
- Le fichier de sortie : c'est un fichier texte qui contient les diffèrents règles générés après l'exécution de l'agorithme approprié, ce dernier contient aussi le support de la règle ainsi le confiance.
- Min-Support : c'est le support minimale saisi par l'utilisateur, pour définir le nombre d'occurrence de chaque partie de la règle.
- Le nombre K : c'est nombre entier consiste à définir l'itervalle de recherche.
- Min-Confiance : c'est la confiance minimale saisi par l'utilisateur pour dire que cette règle est valide.
- Le delta :

Nous présentons cette interface par cet figure ci-après :

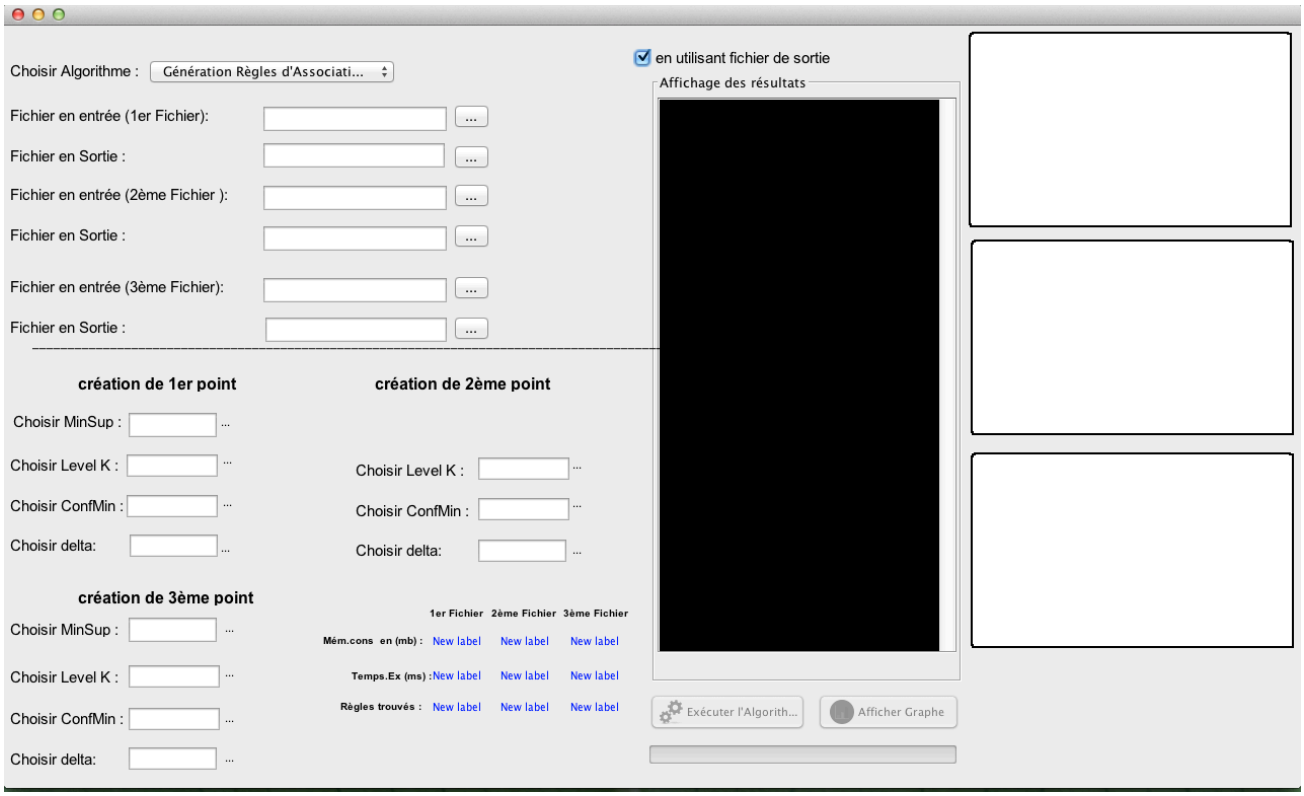


Figure 4.16 : Interface de génération des règles d'association

D'après cette interface, on peut générer plusieurs scénarios pour générer les règles, on va citer quelques scénarios :

1. L'algorithme TopKRules :

   - Nous choisisons l'algorithme **TopKRules** de la liste déroulante en haut, puis nous donnons le chemin des 3 fichiers en entrées ainsi le chemin de sauvegarde de chaque fichier de sortie, qui seras généré automatiquement lors de l'exécution.

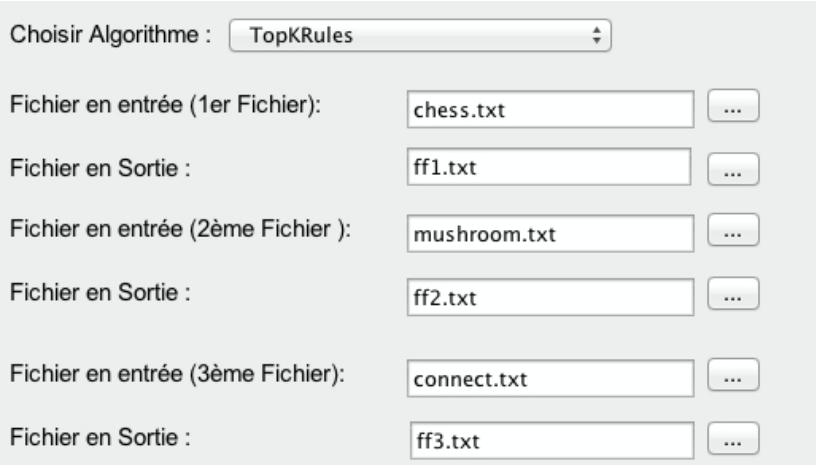


Figure 4.17 : paramètres d'entrée et de sortie

   - En suite, nous introduisons les différents paramètres correspond à l'algorithme : (Nombre k et la confiance minimale).

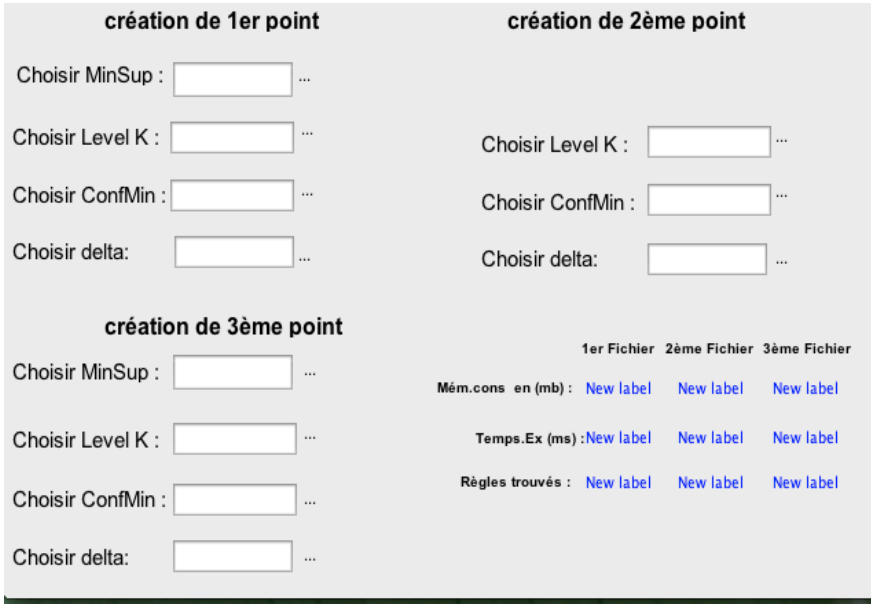


Figure 4.18 : Autres paramètres.

❖ Maintenant, nous cliquons sur le button Exécuter l'algorithme pour voir les résultats obtenus.

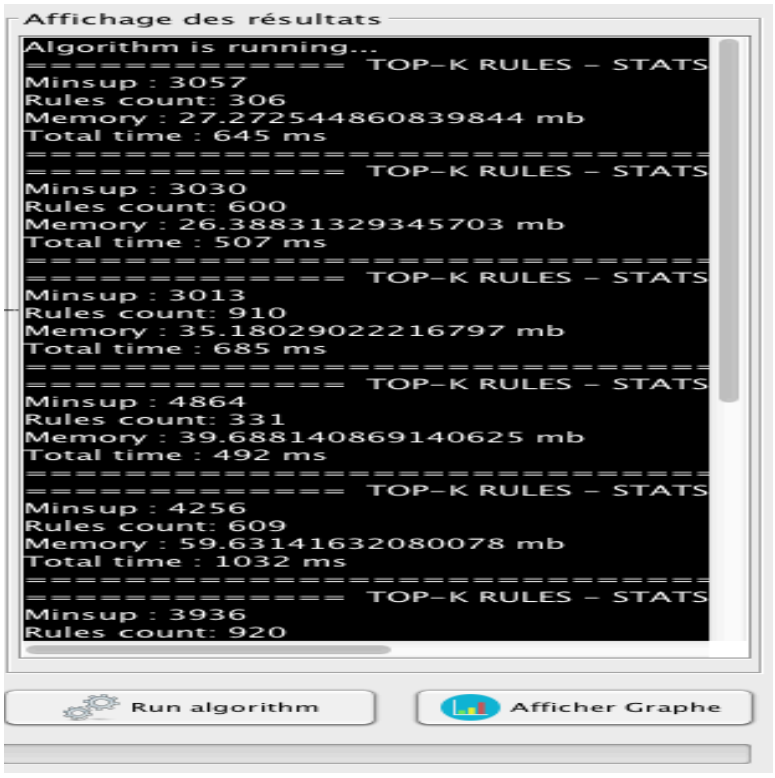


Figure 4.19 : Résultats obtenus Après l'exécution de l'algorithme TopKRules

❖ Ainsi, on peut voire les graphes correspond à ces résultats.

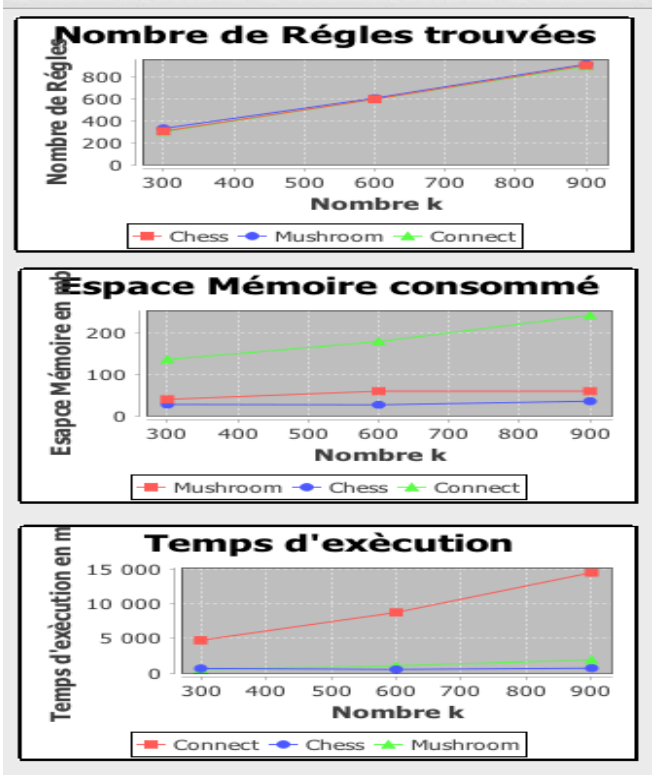


Figure 4.20 : Résultats obtenus présentés par graphes

2. L'algorithme TNR :

❖ Nous choisisons l'algorithme **TNR** de la liste déroulante en haut, puis nous donnons le chemin des 3 fichiers en entrées ainsi le chemin de sauvegarde de chaque fichier de sortie, qui seras généré automatiquement lors de l'exécution.

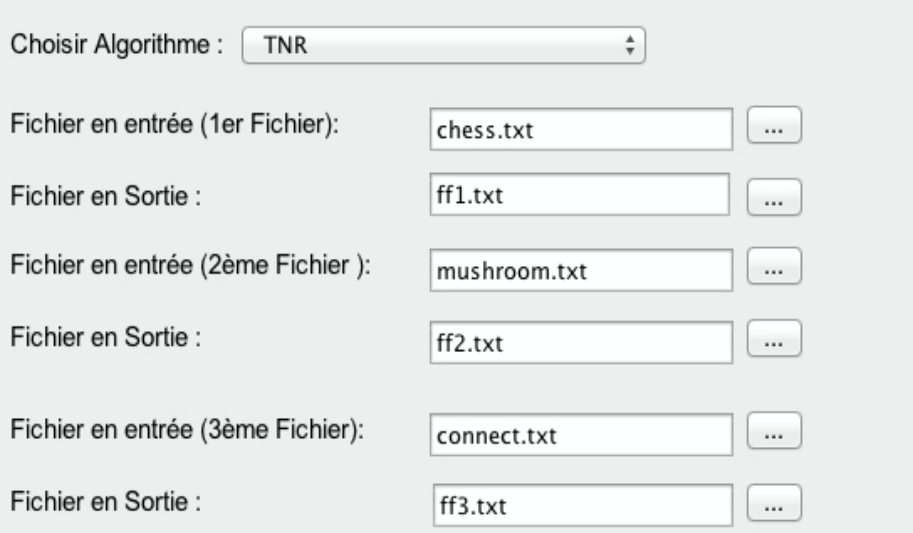


Figure 4.21: paramètres d'entrée et de sortie

❖ En suite, nous introduisons les différents paramètres correspond à l'algorithme : (Nombre k, la confiance minimale et le delta).

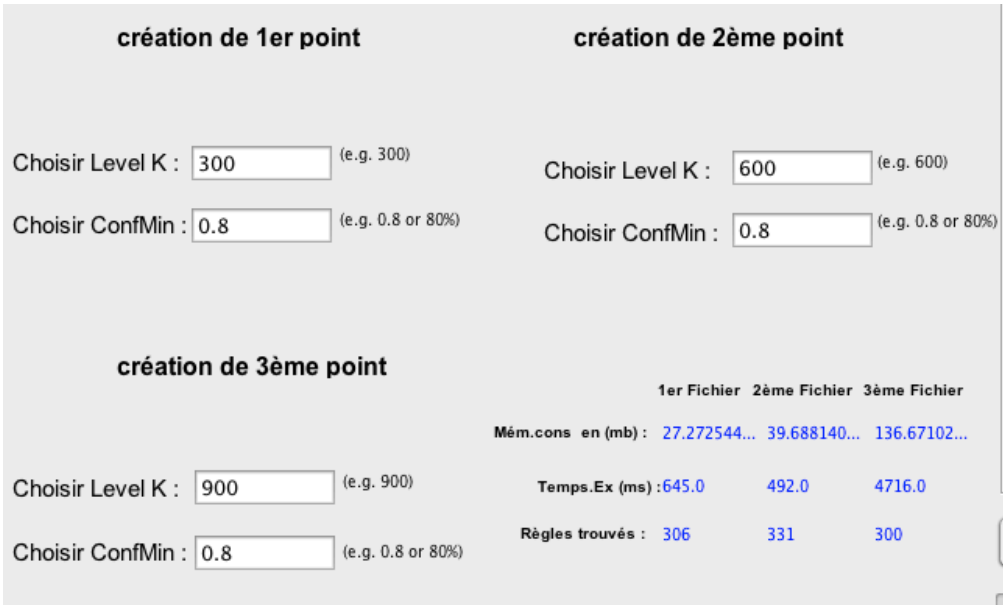


Figure 4.22 : Autres paramètres.

❖ Maintenant, nous cliquons sur le button Exécuter l'algorithme pour voir les résultats obtenus.

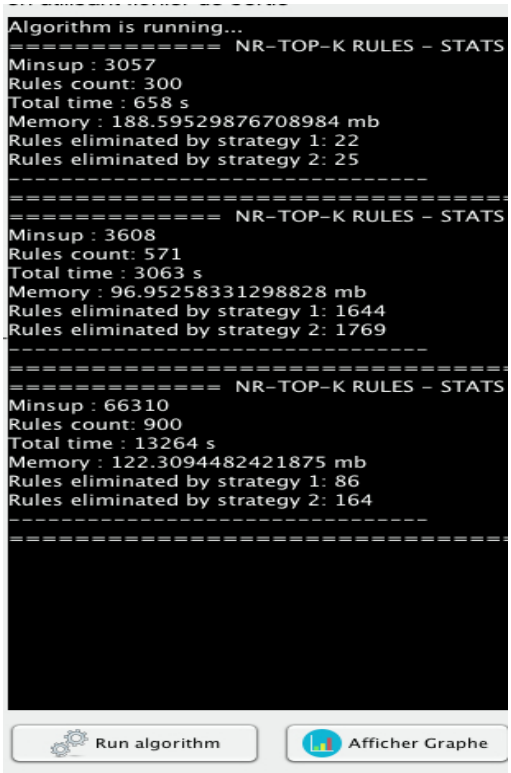


Figure 4.23 : Résultats obtenus Après l'exécution de l'algorithme TNR

❖ Ainsi, on peut voire les graphes correspond à ces résultats.

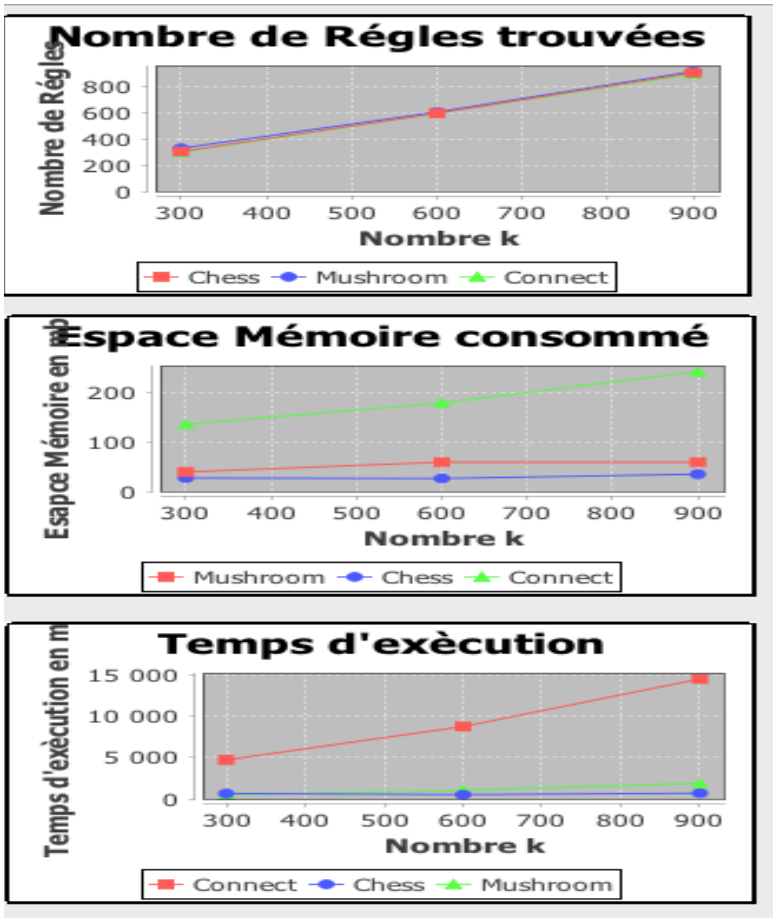


Figure 4.24 : Résultats obtenus présentés par graphes

4.7.2 Description des tables de la base de données :

Dans notre travail, nous avons utilisés plusieurs fichier importés depuis des base de données réels, ils ont été préparé par Roberto Bayardo depuis la platforme UCI et PUMSB [23], parmi ces fichier on trouve :

- Fichier Mushroom.txt (3196 transactions) : Ce fichier de données comprend une description d'échantillons hypothétiques correspondant à 23 espèces de champignons à lamelles dans le Agaricus et Lepiota famille. Chaque espèce est classifié en définitivement comestible, certainement toxiques, ou de la comestibilité inconnue et non recommandée. Cette dernière classe a été combiné avec celui toxique [20].
- Fichier connect.txt (67557 transactions) : Cette base de données contient toutes les positions juridiques 8 plis dans le jeu de connect-4 dans laquelle aucun joueur n'a encore gagné, et dans lequel le prochain mouvement ne soit pas forcé.x est le premier joueur; o le second.La classe de résultat est la valeur théorique de jeu pour le premier joueur[21].

- ❖ Fichier chess.txt (8416 transactions) : ce fichier est généré depuis des situations de jeu entre deux adversaires[23].

4.8 Discussion des Résultats :

Les résultats obtenus après l'exécution des algorithmes plusieurs fois, sont décrit ci-après selon les différents paramètres choisis par l'utilisateur, ces paramètres sont choisis selon les besoins de chaque individu, nous allons détaillés chaque résultats obtenus comme suit :

4.8.1 Algorithme TopKRules pour confiacne = 0,8 :

les résultats sont affichés sur le tableau suivant :

Algorithme Top-K-Rules							
Paramètres/ Donnée	Espace Mémoire consommé en (mb)			Temps d'exécution en (ms)			Confiance =0,8
	k=500	k=2000	k=3500	k=500	k=2000	k=3500	
Chess	28,813	85,914	132,631	846	1174	2161	
Mushroom	195,008	289,330	343,05	1327	4415	6948	
Connect	396,13	362,322	446,054	6952	33150	46840	

Tableau 4.8 : Résultats obtenus après exécution de l'algorithme TopKRules

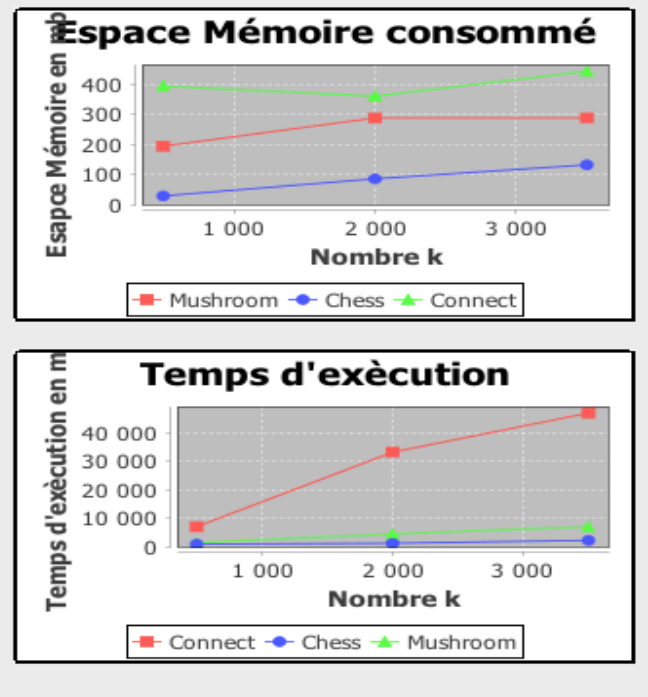


Figure 4.25 : résultats obtenus pour k=500, k=2000 et k=3500 pour chaque fichier de donnée

4.8.2 Algorithme TopKRules pour confiacne = 0,5 :

Les résultats sont affichés sur le tableau suivant :

Algorithme Top-K-Rules							
Paramètres/ Donnée	EspaceMémoire consommé en (mb)			Temps d'exécution en (ms)			
	k=500	k=2000	k=3500	k=500	k=2000	k=3500	
Chess	19,249	28,650	41,924	789	1189	1850	Confiance =0,5
Mushroom	59,94	80,472	120,313	532	1911	3158	
Connect	213,909	334,301	371,390	9096	30398	47342	

Tableau 4.9 : Résultats obtenus après exécution de l'algorithme TopKRules

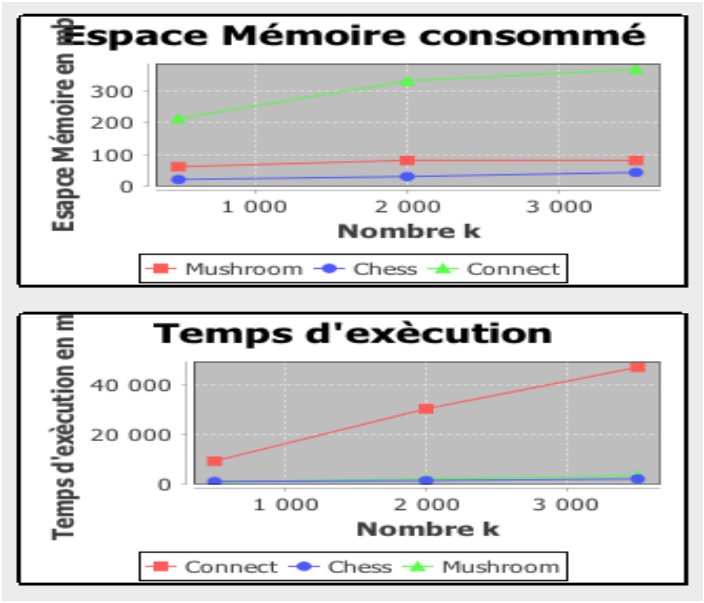


Figure 4.26 : résultats obtenus pour k=500, k=2000 et k=3500

D'après les résultats obtenus, nous constatons les point suivants :

- L'espace mémoire consommé et le temps d'exécution sont rainnoable pour toutes l'ensemble de données.

4.8.3 Algorithme TopKRules Non redunadante pour confiance = 0.5 et delta=200 :

Les résultats sont décrit sur le tableau suivant :

Algorithme Top-K-Rules Non Redondante (TNR)								
Paramètres /Donnée	Espace Mémoire consommé en (mb)			Temps d'exécution en (ms)				
	k=500	k=2000	k=3500	k=500	k=2000	k=3500		
Chess	28,629	32,601	48,582	1109	1473	2553	Confiance =0,5	delta =200<k
Mushroom	247,206	264,724	300,879	3545	14090	27485		
Connect	330,175	366,152	379,992	9702	44096	75429		

Tableau 4.10 : Résultats obtenus après exécution de l'algorithme TopKRules Non redundante

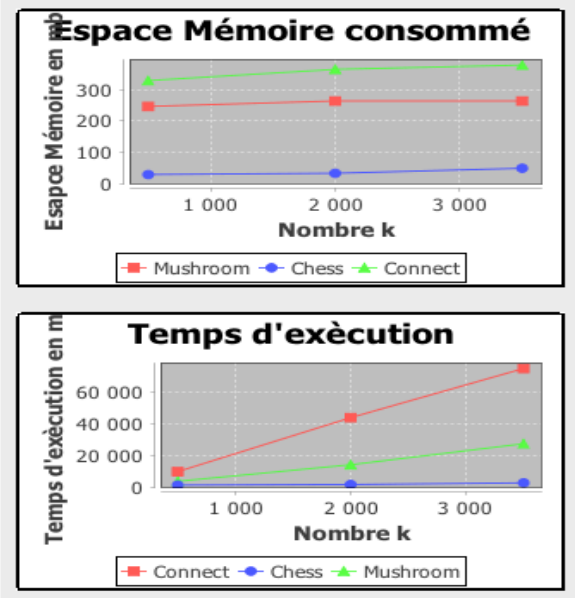


Figure 4.27 : Résultats obtenus pour k=500, k=2000 et k=3500

Ainsi, nous introduisons le nombre des règles eliminés apr les deux stratégies de l'algorithme :

Algorithme Top-K-Rules Non Redondante (TNR)								
Paramètres /Donnée	Règles supprimés par la 1ére stratégie			Règles supprimés par la 2éme stratégie			Confiance =0,5	delta =200<k
	k=500	k=2000	k=3500	k=500	k=2000	k=3500		
Chess	82	407	1006	146	740	1378		
Mushroom	2044	12316	29450	2143	9880	22355		
Connect	53	411	1205	93	735	1956		

Tableau 4.11 : Résultats obtenus après exécution de l'algorithme TopKRules Non redondante pour l'élimination des règles par deux stratégies.

4.8.4 Algorithme TopKRules Non redunadante pour confiance = 0.5 et delta=3600 :

Les résultats sont affichés sur le tableau ci-après :

Algorithme Top-K-Rules Non Redondante (TNR)								
Paramètres /Donnée	Espace Mémoire consommé en (mb)			Temps d'exécution en (s)			Confiance =0,5	delta =3600>k
	k=500	k=2000	k=3500	k=500	k=2000	k=3500		
Chess	48,654	118,77	117,93	4122	4237	5101		
Mushroom	342,054	536,88	431,498	33182	48387	80355		
Connect	518,027	555,790	867,478	90725	125599	183678		

Tableau 4.12 : Résultats obtenus après exécution de l'algorithme TopKRules Non redondante

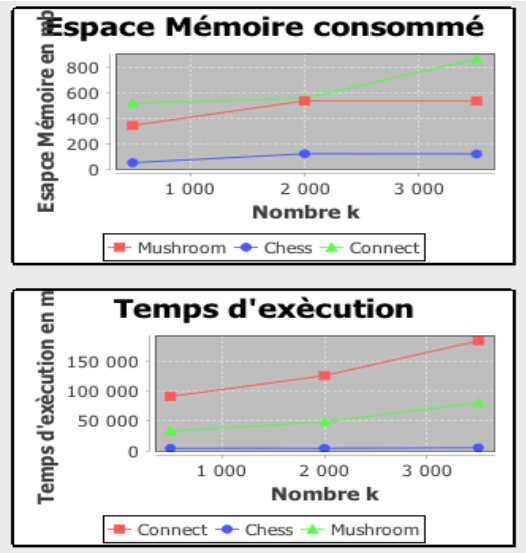


Figure 4.28 : Résultats obtenus pour k=500, k=2000 et k=3500

Algorithme Top-K-Rules Non Redondante (TNR)								
Paramètres/ Donnée	Règles supprimés par la 1ére stratégie			Règles supprimés par la 2éme stratégie			Confiance =0,5	delta =3600>k
	k=500	k=2000	k=3500	k=500	k=2000	k=3500		
Chess	1148	1796	2718	1627	2718	3881		
Mushroom	34004	50530	77751	26588	40112	63106		
Connect	1553	3004	4634	2401	4032	6157		

Tableau 4.13: Résultats obtenus après exécution de l'algorithme TopKRules Non redondante pour l'élimination des règles par deux stratégies.

D'après les résultats obtenus par cet algorithme, Nous observons les points suivants :

- L'espace mémoire et le temps d'exécution sont obtenus d'une façon exacte et précis par rapport aux paramètres selectionner par l'utilisateur
- Cet algorithme utilise trop de mémoire prcequ'il génère un grand espace de recherche.
- Les stratégies d'elmination des règles permet de diminuer le redondance.

4.8.5 Fichiers de sortie :

Après chaque exécution de l'algorithme, des fichiers de sortie seront générées d'une façon automatique, on va voire un exemple de fichier ci-après :

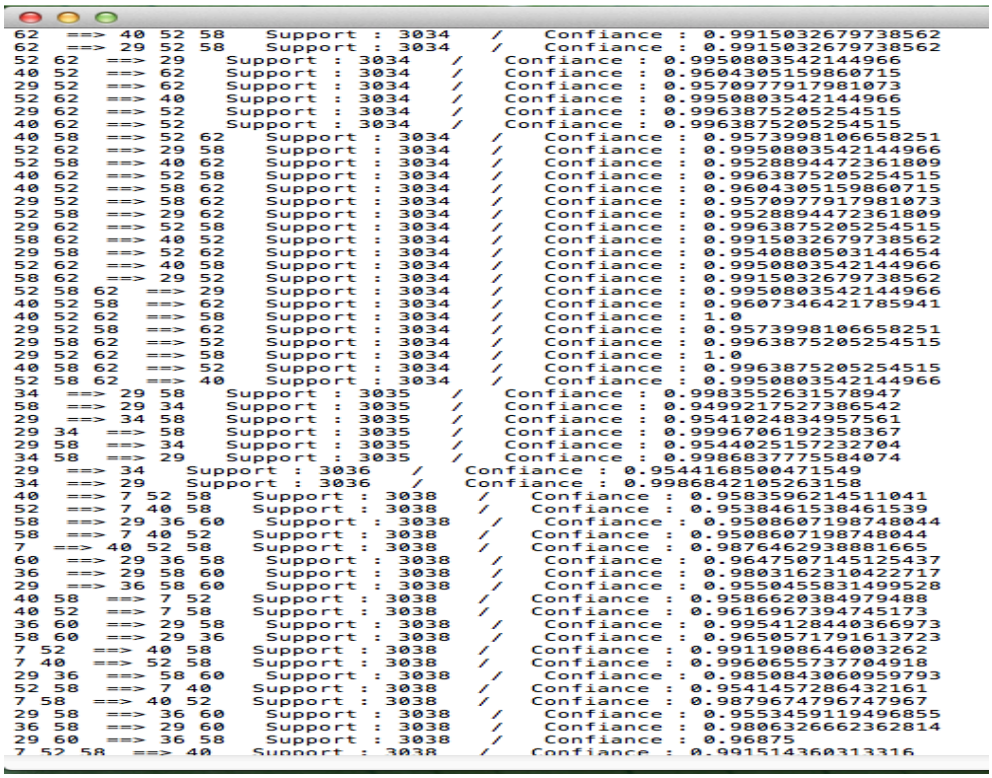


Figure 4.29 : fichier de sortie sous format de texte

Ce fichier présente les différents règles générées après l'xécution de l'algorithme approprié, il contient trois (3) parties essentielles :

- ❖ Partie règle d'association qui contient une antécédent (partie gauche) et un conséquent (partie droite
- ❖ Le supprot de de la règle d'assocition
- ❖ Le confiance de la règle.

4.9 Comparaison entre les deux algorithmes (TopKRules et TNR) :

En se basant sur les résultats obtenus précédèment, on peut établir une comparaison entre les deux algorithmes TopKRules et TNR selon les paramètres étudier, le tableau décrit ci-après montre la diffrénce entre les deux algorithme pour k=3600 et confiane = 0,5

Paramètres/ Donnée	Espace Mémoire en mb		Temps d'exécution en ms			
	TopKRules	TNR	TopKRules	TNR		
Chess	41,924	117,93	1851	5101	Confiance =0,5	K=3500
Mushroom	120,313	431,498	3158	80355		
Connect	371,390	867,478	47342	183678		

Tableau 4.14: comparaison entre les deux algorithmes.

D’après ces résultats on peut dire le suivant :

- L’algorithme TNR est trop coûteux en terme de temps d’exécution et espace mémoire, donc il génère un espace de recherche grand pour générer les règles d’association.
- L’algorithme TNR deminue le taux de la redondance par l’ajout d’un paramter (delta) à l’espace de recherche
- L’algorithme TNR donne une résultat exact par l’augmentation de l’espace de recherche des régles
- Pour l’algorithme TopKRules, le temps d’exécution et l’espace mémoire sont raisonable mais le taux de la redondance est grande.
- Le temps d’exécution de l’lagorithme TopKRules possède une meilleur evolutivité par rapport au nombre k.

4.10 Conclusion :

Dans ce chapitre, nous avons vu les différents outils utilisés pour implémenter notre application, ainsi les interfaces de l’application, ensuite nous introduisons tous les résultats obtenus après l’exécution de plusieurs fois de chaque algorithmes étudier.