Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

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 Master en

Informatique

Option: Réseau et Système Distribuée (RSD)

Thème

Conception et réalisation d'une application client serveur en utilisant le Middleware JAVA RMI (gestion des absences)

Réalisé par :

- **HAMIMED Yazid**
- **HAMIMED Soumia (nee BENDAHMANE)**

Présenté le 03 Juliet 2017 devant le jury composé de MM.

- **BELHABI Amel (Encadrant)**
- **MANA Mohamed (Président)**
- **BENMOUNA Youcef (Examinateur)**

Remerciements

En préambule à ce mémoire nous remercions ALLAH qui nous a aidé et nous a donné la patience et le courage durant ces langues années d'étude.

Ces remerciements vont tout d'abord aux nos parents. Vous trouverez ici le résultat de longues années de sacrifices et de privations. Merci pour les valeurs nobles, l'éducation et le soutien permanent venu de vous. Et nos enseignants du département d'informatique pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts et une grande technique pédagogique pour assurer à leurs étudiants une formation actualisée.

Nous souhaitons adresser encore nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Nous tenons à remercier très sincèrement Madame BELHABI Amel qui, en tant qu'encadrer de notre mémoire, s'est toujours montré à notre écoute et très disponible tout au long de la réalisation de ce mémoire. Ainsi nous lui devons beaucoup pour sa contribution, son aide et ses conseils en consacrant son temps pour que ce projet de fin d'étude réussisse à son optimum. . Merci de nous avoir accompagnés tout au long de ce projet avec beaucoup de rigueur scientifique et de précieux conseils et remarques qui nous ont permis de réaliser ce travail.

Nous remercions les membres du jury d'avoir pris le temps de lire et d'évaluer notre travail.

Nous tenons encore à exprimer nos sincères remerciements à tous les professeurs qui nous ont enseigné et qui par leurs compétences nous ont soutenu dans la poursuite de nos études.

Enfin, nous remercions toute personne qui a participé de près ou de loin pour l'accomplissement de ce modeste travail.

DÉDICACE

Toutes les lettres ne sauraient trouver les mots qu'il faut… Tous les mots ne sauraient exprimer la gratitude, L'amour, le respect, la reconnaissance… Aussi, c'est tout simplement que je dédié cette mémoire

A l'âme de mon cher père, source de mon inspiration dans la vie,

A ma chère mère pour son soutien,

A Mon frères et ma sœur et leur enfants .

Je dédie aussi ce travail à ma femme qui m'a toujours encouragée dans mes études et m'a préparé le climat favorable d'avancement.

Ce mémoire est aussi dédié à mon adorable bébé Sidi Mohamed.

Yazid

DÉDICACE

C'est avec l'immense plaisir et le grand honneur que je dédie ce mémoire :

À mes chères parents qu'ils méritent tout le bonheur du monde et je leurs dis « Vous avez tout sacrifié pour vos enfants n'épargnant ni santé ni efforts. Vous m'avez donné un magnifique modèle de labeur et de persévérance merci».

À mon chère mari «Yazid » et je lui dis « je tiens à exprimer toute ma reconnaissance et un immense remerciement à toi pour ta confiance en moi et ta patience avec moi merci ».
À mon petit bébé « Sidi Mohamed » que DIEU me protège et je leur dis tu es ma force

À mes chères frères « Yousef », « Zakaria » et à ma charmante sœur « Asmaa » et je leurs dis Je vous aime très fort.

À ma belle famille: ma belle mère que j'aime beaucoup, mon beau père, ma belle sœur « Fayza » mon beaux frère « Ahmed » Et leur femme attrayante.
et bien sûr à les adorables enfants Mohamed,Kawter,Ibrahim,Rayan,Saja,Yakine,Rassim

À tout membre de ma grande famille: tantes, oncles, cousins et cousines.
À mes enseignants de la filière Informatique.

À tous ceux qui ont contribué de près ou de loin à l'édition de ce mémoire.

Soumia

Sommaire

INTRODUCTION GENERALE 10

I. Chapitre 1 Architecture Client -Serveur & java RMI 14

I.1 Introduction 14

I.2 Systèmes distribués 14

I.2.1 Définition 14

I.2.2 Les types d'architecture de système distribués 14

I.3 Architecture client serveur 15

I.3.1 Définition 15

I.3.2 Le serveur 15

I.3.3 Les clients 16

I.3.4 Une requête 16

I.3.5 Réponse 16

I.3.6 Avantages de l'architecture client/serveur 16

I.3.7 Inconvénients du modèle client/serveur 16

I.3.8 Les différentes architectures client/serveur 17

I.3.9 Les modes d'interaction client/serveur 20

I.4 Notion de middleware 20

I.5 Remote Method Invocation (RMI) 21

I.5.1 Introduction 21

I.5.2 Définition 21

I.5.3 L'application RMI 21

I.5.4 Package RMI 21

I.5.5 Quelques classes 22

I.5.6 Architecture RMI 22

I.5.7 Principe de fonctionnement RMI 24

I.5.1 Déploiement de RMI 25

I.6 Conclusion 27
II. Chapitre 2 : Analyse & Conception 29
II.1 Introduction 29
II.2 Etude de l'existant 29
II.3 Problématiques 30
II.4 Spécification des besoins 31
II.4.1 Besoins fonctionnels 31
II.4.2 Besoins non fonctionnels 32
II.5 Conception du système 33
II.6 Le langage de modalisation UML 33
II.7 ArgoUML 34
II.8 Identification des acteurs 34
II.9 Identification des activités 34
II.10 Diagramme de cas d'utilisation 35
II.10.1 Identification des acteurs 35
II.10.2 Identification des cas d'utilisations 35
II.11 Spécification des tâches et scenarios du système 37
II.12 Diagramme de séquence 37
II.13 Diagramme de classes 39
II.1 Conclusion 40
III. Chapitre 03 Implémentation 41
III.1 Introduction 41
III.2 Java 41
III.3 Eclipse 43
III.4 JDBC (java database connectivity) 43
III.5 EasyPHP et MySQL 45
III.6 Présentation de l'application 45
III.7 Développement de gestion d'absences 46
III.8 Application gestion des absences 46

III.9 La sécurité dans notre application 55
III.10 Conclusion 56
Conclusion Générale 57
Bibliographies 59

Liste des tableaux

Tableau 1 : Spécification des tâches et scenarios du système 37

Liste des figures

Figure 1 Architecture client-serveur 15
Figure 2 : Présentation de l'architecture 3-tiers 18
Figure 3 : Les architectures 3-tiers 19
Figure 4 : Les architectures n-tiers 20
Figure 5 : Structure des couches RMI 22
Figure 6 : Principe de fonctionnement RMI 25
Figure 7 : Déploiement de RMI 26
Figure8 : Méthodologies de modélisation 33
Figure 9 : Classification Des Diagramme UMl 33
Figure 10 : Diagramme de cas d'utilisation 36
Figure 11 : Diagramme de séquence (exemple d'authentification) 38
Figure 12 : Diagramme de séquence (exemple d'ajouter absence) 39
Figure 13 : Diagramme de Classe 40
Figure 14 : Fenêtre d'authentification 48
Figure 15 Vérification de mot de passe 48
Figure 16 : indication de mot de passe 49
Figure 17 : Menu principal serveur 49
Figure 18 : gestion des étudiants 50
Figure 19 : Fenêtre d'authentification agent 51
Figure 20 : Le menu cote client 51
Figure 21 : Ajouter les absences 52
Figure 22 : Modifier absence 53
Figure 23 : Résultat de recherche par date/module 54
Figure 24 : Afficher les étudiants Exclus 54
Figure 25 : consultation des justificatifs 55
Figure 26 : imprimer la liste de présence 61
Figure 27 : la partie1 du questionnaire 62
Figure 28 : la partie2 du questionnaire 63
Figure 29 : Aperçu de la création d'un réseau ad hoc via le Centre Réseau et partage 64
Figure 30 : Aperçu d'assistant de création d'un réseau ad hoc 64
Figure 31 : Aperçu d'informations relatives à la création d'un réseau poste à poste 64
Figure 32 : Aperçu de configuration de la sécurité du réseau ad hoc 64

INTRODUCTION GENERALE

Le phénomène informatique qui n'est plus exclusivement un outil de bureautique, s'est progressivement et rapidement répandu dans tous les domaines, notamment dans le domaine de la gestion, où son utilisation a été longtemps appréhendée.

De nos jours, dans le milieu de la scolarité, l'informatique prend petit à petit une place importante et grandissante, ainsi la gestion des opérations scolaire devient beaucoup plus sûre et efficace ce qui a suscité l'intérêt d'un bon nombre d'établissements pour cette informatique plus « moderne ».

L'évolution des langages de programmation a amené de nouveaux outils aidant à la conception des applications de gestion en informatique. En effet, l'événement de l'orienté objet facilite l'abstraction du problème à résoudre en fonction des données du problème lui-même (par l'utilisation des classes et d'objets).

C'est ainsi, que la réalisation de logiciels de gestion demeure une activité professionnelle difficile. Malgré les progrès apportés par le génie logiciel, les développements d'application répondant aux besoins exprimés se rationalisent lentement. En revanche, l'offre d'outils de développement ne cesse de croitre et l'importance des langages de programmation est toujours prépondérante. De plus en plus d'applications utilisent même plusieurs langages de programmation dans le cadre d'un projet unique. Nous constatons enfin, un intérêt grandissant pour issus des technologies objets.

Nous nous intéressons au service de scolarité de la faculté des sciences d'université Abou Bakr Belkaid de Tlemcen. Cette faculté contient quatre filières Informatique, Mathématique, Physique, Chimie. Dans chaque filières il ya plusieurs options alors un membre important des étudiants.la gestion des absences de ces derniers devrai très difficile, un enseignant fait un appel pour chaque séance sur un fichier Excel ou même sur papier qui rend l'algorithme de sauvegarde un peut délicat. Ainsi quand un étudiant prouver son absences par une justificatif les enseignant trouve des difficultés de mentionner et de l'archiver et de consulter après en cas ou ils sont besoin de la vérification, aussi l'action de recherche des absences est presque impossible, donc la gestion manuel des absences des étudiant engendre un risque de perte une grande quantité d'information avec un énorme gaspillage du temps.

De ce fait notre faculté est besoin a une plateforme informatisée prenant en compte toutes les données de manière systématisée améliorerait la rentabilité de le stockage et une gestion efficace réduirait le risque d'erreur et le temps des traitements.

C'est donc nous essayant de rénover la gestion des absences que l'informatisation de la pratique s'impose ; l'instauration d'une base de données numérique est une nécessité

exprimée par une équipe pédagogique. Ce système accompagne les enseignants, les chefs de département et les secrétaires dans ses pratiques quotidiennes pour améliorer la gérance des absences des étudiants. L'application qui enregistre tous les absences par date, module et séance pour chaque étudiant, de plus l'application permet de sauvegarder leurs justificatifs et de les alerter s'ils sont proche d'être exclus.

L'objectif de notre PFE est faire la gestion des absences en se basant sur une architecture de type client/serveur et en utilisant le mécanisme JAVA RMI.

Notre rapport de mémoire est organisé comme suit :

- Le premier chapitre présente des généralités l'architecture client serveur et le middleware java RMI
- Le deuxième chapitre est consacré à l'analyse et à la conception de notre système.
- Le troisième chapitre est dédié à l'implémentation de notre application avec les outils de développement et les langages de programmation.

Chapitre 1 Architecture Client - Serveur & java RMI

I. Chapitre 1 Architecture Client -Serveur & java RMI

I.1 Introduction

C'est en 1994 que l'architecture client-serveur s'implante sur le marché, car Il devint nécessaire de créer une architecture de communication dans le but de centraliser les informations, ainsi que de les sécuriser le mieux possible.

L'architecture client-serveur contribue à augmenter considérablement la rapidité de traitement des postes de travail grâce a la structure du serveur qui contient plusieurs unités de traitement (CPU). La capacité ainsi que la mémoire sont accrues. La technologie client-serveur se développera à un rythme des plus accélérés On peut affirmer qu'elle a connu une croissance exponentielle dans tous les domaines d'activités, nous citons :

- Gestion de base de données.
- Les systèmes transactionnels.
- Les systèmes de messagerie, web, Internet
- Les systèmes de partages des données.

I.2 Systèmes distribués

I.2.1 Définition

Un système distribues est ensemble d'ordinateurs indépendants connectés en réseau et communiquant via ce réseau .Cet ensemble apparaît du point de vue de l'utilisateur comme une unique entité. (Andrew Tannenbaum) [1]

Vision matérielle d'un système distribué : architecture matérielle Machine multiprocesseurs avec mémoire partagée, CPU multicore Cluster d'ordinateurs dédiés au calcul/traitement massif parallèle , Ordinateurs standards connectés en réseau

Vision logicielle d'un système distribué : système logiciel composé de plusieurs entités logicielles s'exécutant indépendamment et en parallèle sur un ensemble d'ordinateurs connectés en réseau

« Un système distribué est un système qui m'empêche de travailler quand une machine dont je n'ai jamais entendu parler tombe en panne » Leslie Lamport.

I.2.2 Les types d'architecture de système distribués

- **Systèmes distribués** : coopération de systèmes sans objectif figé.

- **Cluster** : grappe de machine homogènes localisées (partage de ressource, ripartition des charges, disponibilitees)
- **Systèmes fédérés :** coopération de systèmes avec un objectif commun.
- **Grid** : infrastructure constituée d'un ensemble coordonné de ressources potentiellement partagée, distribuées, hétérogène et sans administration centraliser.[2] [3]

I.3 Architecture client serveur

I.3.1 Définition

L'architecture client-serveur est une architecture logicielle dans laquelle les programmes d'application, dits clients, font appel, dans le cadre d'un réseau, à des services génériques distants fournis par des ordinateurs appelés serveurs.

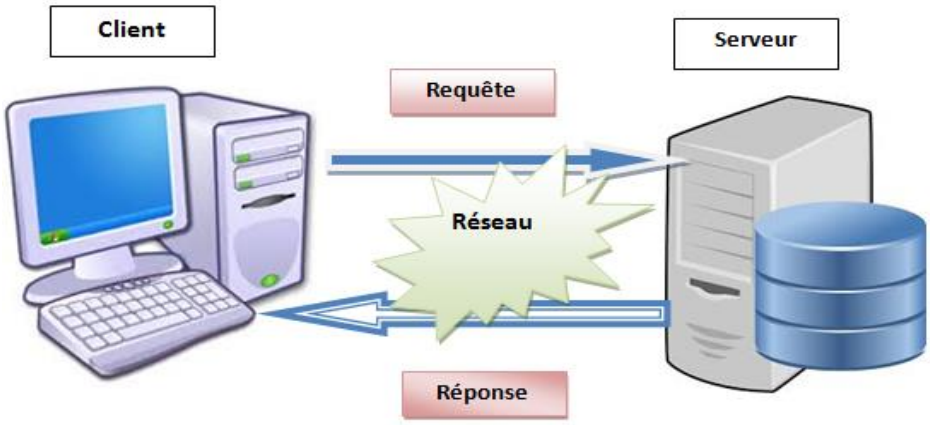


Figure 1 Architecture client-serveur

I.3.2 Le serveur

Ordinateur spécialisé dans la fourniture et le stockage des ressources partagées des utilisateurs réseau.

Caractéristiques d'un serveur : Un serveur possède les caractéristiques suivantes :

- Il est fournisseur de service.
- Il est à l'écoute et prêt à répondre aux requêtes envoyées par des clients.
- Dès qu'une requête lui parvient, il la traite et envoie une réponse à son expéditeur.

- Il est capable de Traiter plusieurs clients simultanément.

I.3.3 Les clients

Ce sont des terminaux qui accèdent aux ressources partagées fournies par un serveur du réseau.

Caractéristiques d'un client: Le client à son tour possède des caractéristiques parmi lesquelles nous citons :

- C'est un consommateur de service.
- Il établit une connexion au serveur.
- Il envoie des requêtes au serveur.
- Il attend et reçoit les réponses du serveur.

I.3.4 Une requête

Message transmis par un client à un serveur décrivant l'opération à exécuter pour le compte du client.

I.3.5 Réponse

Message transmis par un serveur à un client suite à l'exécution une opération, contenant le résultat de l'opération.

Le client et le serveur doivent bien sûr utiliser le même protocole de communication.

Dans une architecture client-serveur on trouve d'un côté le client et de l'autre le serveur. Cette d'architecture peut se faire sur tout type d'architecture matérielles interconnectées. Concrètement,

Le client demande un service au serveur, le serveur reçoit cette requête effectue un traitement, et renvoie la ressource demandée par le client. [4]

I.3.6 Avantages de l'architecture client/serveur

Le model client serveur fournit un grand niveau de fiabilité, ses principaux atouts sont :

- **Une administration centralisée** : l'administration de tout le système est centralisée au niveau du serveur.
- **La sécurité** : car le nombre de points d'entrée permettant l'accès aux données est moins important.
- **Une gestion au niveau serveur** : les clients sont administrés par le serveur, ce dernier est le seul responsable de leur gestion.

I.3.7 Inconvénients du modèle client/serveur

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

- **La pane :** une pane au niveau du serveur paralyse tout le réseau.
- **Un coût élevé** : dû à la technicité du serveur.

I.3.8 Les différentes architectures client/serveur

Une application informatique peut être découpée en trois niveaux d'abstraction distincts:

La couche de présentation : permet l'interaction de l'application avec l'utilisateur. Cette couche gère les saisies au clavier, à la souris et la présentation des informations à l'écran.

La logique applicative, les traitements: écrivant les travaux à réaliser par l'application. Ils peuvent être découpés en deux familles :

Les traitements locaux : regroupant les contrôles effectues au niveau du dialogue avec l'IHM, visant essentiellement le contrôle et l'aide à la saisie.

Les traitements globaux: constituant l'application elle-même, contient les règles internes qui régissent une entreprise donnée.

- **Les données :** l'accès aux données, regroupant l'ensemble des mécanismes permettant la gestion des informations stockées par l'application**.**

Ces trois niveaux peuvent être imbriqués ou repartis de différentes manières entre plusieurs machines physiques.

Le découpage et la répartition de ce noyau permet de distinguer les architectures applicatives suivantes

- **L'architecture 1-tiers**
- **L'architecture 2-tiers**
- **L'architecture 3-tiers**
- **Les architectures n-tiers**

I.3.8.1 Présentation de L'architecture 1-tiers

Dans une application un tiers les couches applicatives sont liées et s'exécutent sur le même ordinateur. On ne parle pas ici d'architecture client-serveur, mais d'informatique centralisée. Dans ce contexte plusieurs utilisateurs se partagent des fichiers de données stockés sur un serveur commun.

I.3.8.2 Présentation de l'architecture 2-tiers

L'architecture à deux niveaux (aussi appelée architecture 2-tiers) caractérise les systèmes clients/serveurs pour lesquels le client demande une ressource et le serveur la

lui fournit directement, en utilisant ses propres ressources. Cela signifie que le serveur ne fait pas appel à une autre application afin de fournir une partie du service.

L'échange de messages transite à travers un réseau reliant les deux machines (client et serveur), il met en œuvre des mécanismes relativement complexes qui sont, en général, pris en charge par un middleware.

L'architecture 2-tiers présente de nombreux avantages qui lui permettent de présenter un bilan globalement positif : elle permet l'appropriation des applications par l'utilisateur, l'utilisation d'une interface utilisateur riche et elle introduit la notion d'interopérabilité.

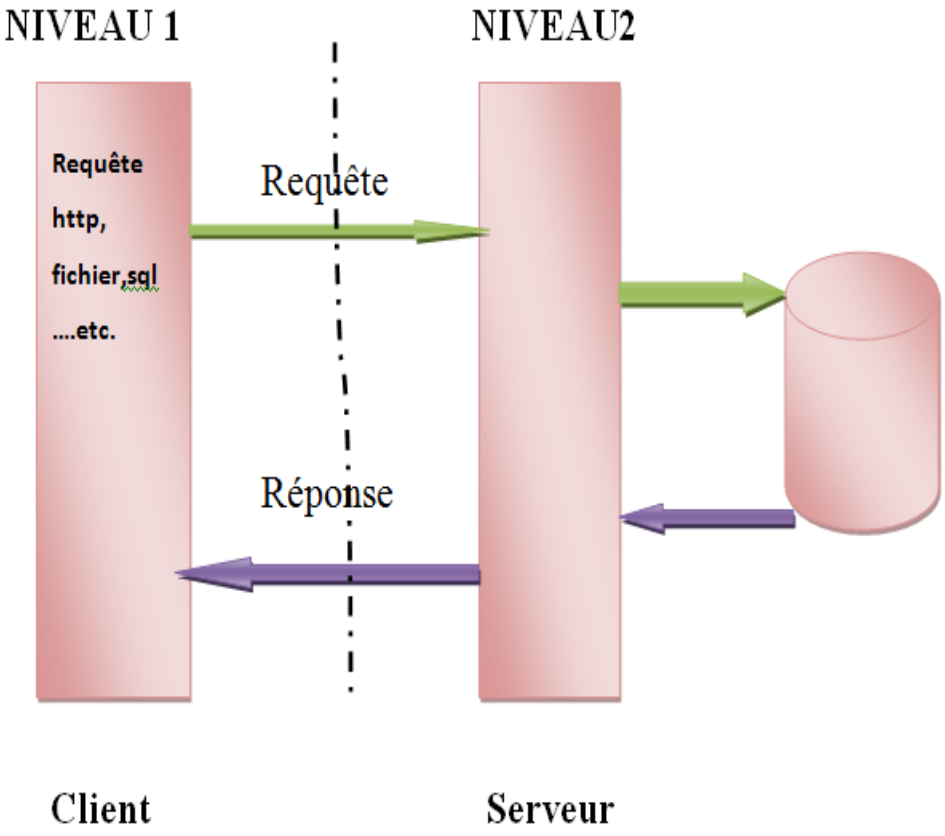


Figure 2 : Présentation de l'architecture 3-tiers

I.3.8.3 Présentation de l'architecture 3-tiers

Cette architecture 3-tiers également appelée client-serveur de deuxième génération ou client-serveur distribué applique les principes suivants :

Les données sont toujours gérées de façon centralisée, la présentation est toujours prise par le post client, la logique applicative est prise en charge par un serveur intermédiaire.

L'architecture 3-tiers sépare l'application en 3 niveaux de services distincts, conforme aux principes précédents :

- **Premier niveau** : l'affichage et les traitements locaux (contrôle de saisie, mise en forme de données…etc.) sont pris en charge par le post 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.

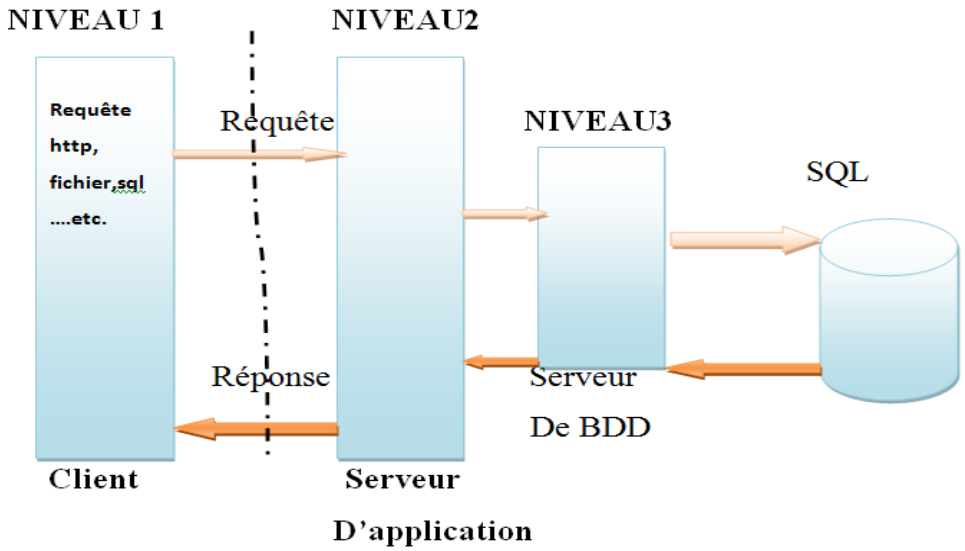


Figure 3 : Les architectures 3-tiers

I.3.8.4 Les architectures n-tiers

L'architecture multi-niveaux a été pensée pour pallier aux limitations des architectures trois tiers et concevoir des applications puissantes et simples à maintenir. Elle qualifie la distribution d'application entre de multiples services et non la multiplication des niveaux de services.

L'architecture n-tiers supprime tous les inconvénients des architectures précédentes :

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

- Elle est basée sur la programmation d'objet ainsi sur des communications standard entre application et le concept Middleware objet [5]

1.

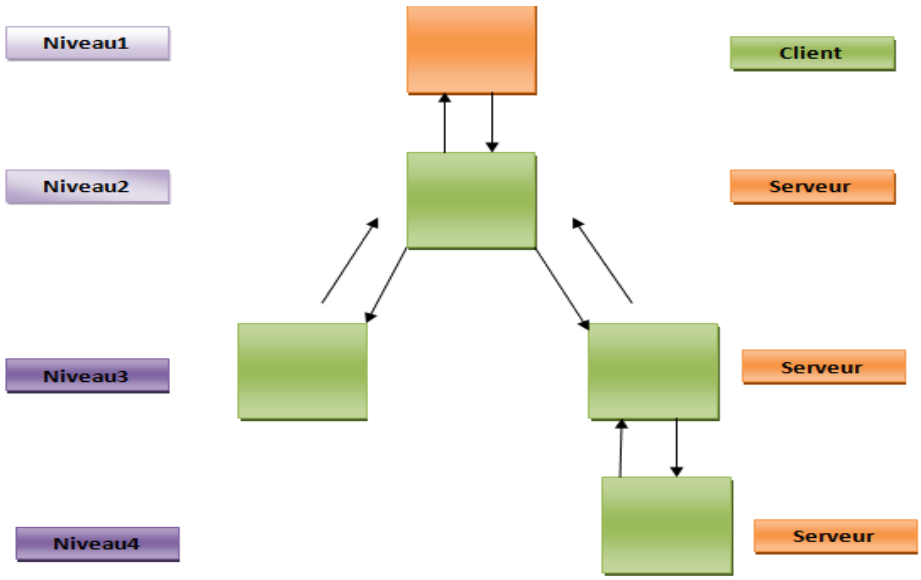


Figure 4 : Les architectures n-tiers

I.3.9 Les modes d'interaction client/serveur

Client serveur de donnée : le modèle C/S de données regroupe les protocoles qui ont pour objectif le transfert de données, SMB, FTP, NFS, JDBC, ODBC, POP etc..

Le modèle C/S de d'affichage regroupe les protocoles qui ont pour objectif de permettre la prise de contrôle à distance TELNET, SSH, X11, ICA, VNC, RDP, etc.

Client de procédures : Le modèle C/S de procédures regroupe les protocoles qui ont pour objectif l'exécution à distance RPC, Java-RMI (JRMP), CORBA-IIOP. [6]

I.4 Notion de middleware

Le middleware est une couche intermédiaire (couche logiciel) qui s'intercale entre l'infrastructure de communication d'un réseau et les éléments de l'application distribuée.

Un Middleware est un logiciel de communication qui permet à plusieurs processus s'exécutant sur une ou plusieurs machines d'interagir a travers un réseau. JM Allio

Un middleware (intergiciel) désigne un ensemble de logiciels se plaçant au dessus du système d'exploitation et servant d'intermédiaire entre les différents composants logiciels d'un système d'information.

On utilise généralement du middleware comme intermédiaire de communication entre des applications distribuées (programmes applicatifs, BD, capteurs etc.).

Exemples : CORBA, JAVA-RMI sont des middlewares orienté traitement permettant l'appel de méthodes à distance, JBDC est un middleware orienté données. [7]

I.5 Remote Method Invocation (RMI)

I.5.1 Introduction

Après que le langage Java a été proposée comme un langage universelle, une question intéressante a été soulevée: comment régler Le problème de l'hétérogénéité: les différences entre les processeurs, les systèmes d'exploitation, les langues et les formats de données?

Le Java Remote Method Invocation (RMI) API est à la fois une démonstration active de la réponse et un Outil populaire pour la construction de systèmes distribués.L'API Java RMI a été conçue à l'origine par Ann Wollrath, Roger Riggs et Jim Waldo au Sun.Avec l'évolution des langages de programmation et de l'orienté objet, la communication client-serveur s'est encore améliorée et rendue plus accessible.

I.5.2 Définition

RMI est un ensemble de classes qui permettent la communication entre machines virtuelles Java (JVM) qui peuvent se trouver physiquement sur la même machine ou sur deux machines distinctes: c.à.d. la manipulation des objets sur des machines distantes (objets distants) de manière similaire aux objets sur la machine locale (objet locaux).

RMI ajoute au java la puissance et la flexibilité de l'appel de procédure à distance (RPC Remote Procedure Calls) dans le cas orienté objet,

I.5.3 L'application RMI

Une application RMI est composée d'une partie client et d'une partie serveur. Le rôle d'un serveur est de créer des objets qu'on qualifie de « distants », de les rendre accessibles à distance et enfin d'accepter des connexions de clients vers ces objets.

Le rôle d'un client est donc d'accéder aux méthodes de ces objets, tout en considérant comme si ces objets étaient des objets locaux. RMI est très souvent utilisé en parallèle avec l'API d'annuaire JNDI (services de nommage). afin que les clients trouvent les services distants, ceux-ci doivent au préalable être enregistrés dans un annuaire, pour cela RMI fourni un rmiregistry.

rmiregistry : possède une table de hachage dont les clés sont des noms et les valeurs sont des objets distants. [8]

I.5.4 Package RMI

C'est un système d'objets distribués constitué uniquement d'objets Java, et qui permet et assure la communication entre machines virtuelles Java.

I.5.5 Quelques classes

java.rmi.Naming.

java.rmi.Remote.

java.rmi.RemoteException.

java.rmi.server.UnicastRemoteObject.

java.rmi.registry.LocateRegistry.

java.rmi.NotBoundException.

I.5.6 Architecture RMI

L'architecture RMI définit la manière dont se comportent les objets, comment et quand des exceptions peuvent se produit, comment gérer la mémoire et comment les méthodes appelées passent et reçoivent les paramètres.

Le système RMI contient 3 couches qui sont :

- **La couche des amorces (stub/skelton).**
- **La couche des références (RRL).**
- **La couche de transport.**

Chacun de ses couches est indépendante de l'autre et utilise un protocole spécifique, la transmission des objets utilise deux technique :

- La sérialisation
- Le chargement dynamique du stub, permet au client de charger dynamiquement le stub quand il a seulement l'interface.

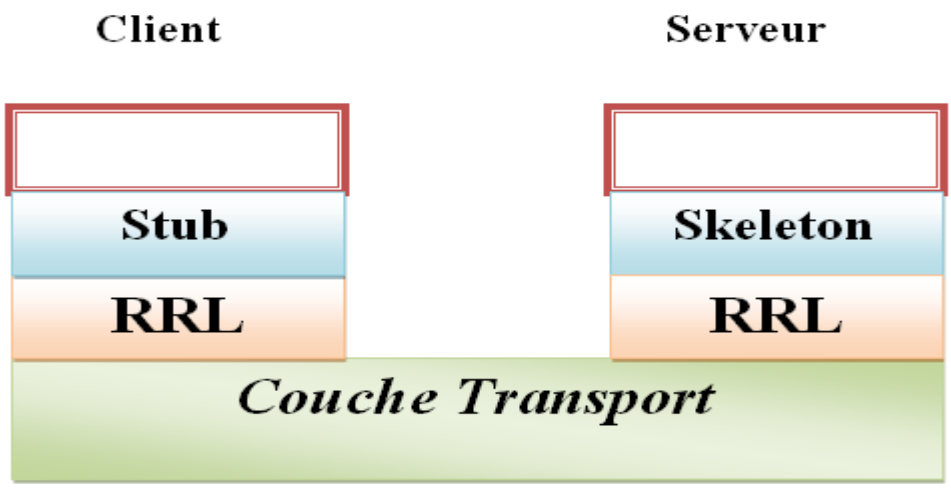


Figure 5 : Structure des couches RMI

- **1ère couche : Stub/Skeleton**

Le stub (traduisez souche) et le skeleton (traduisez squelette), respectivement sur le client et le serveur, assurent la conversion des communications avec l'objet distant.

Stub (coté client) : est un mandataire de l'objet qui est chargé dans le client au moment de l'obtention de la référence, cette référence implémente les mêmes interfaces que l'objet distante, le stub a pour but de :

- Initie une connexion avec la JVM distante en transmettant l'invocation distante à la couche des références d'objets (RRL).
- Assemble les paramètres pour leur transfert à la JVM distante.
- Attend les résultats de l'invocation distante, désassemblent la valeur ou l'exception renvoyée, et renvoient la valeur à l'appelant.

Skeletons (coté serveur) : contient une méthode qui appelle les méthodes de l'objet distant.

- Il a pour but de :
- Désassemblent les paramètres pour la méthode distante.
- Font l'appel à la méthode demandée.
- Assemblage du résultat (valeur renvoyée ou exception) à destination de l'appelant.
- **2ème couche : RRL (couche des références d'objets)**

La couche de référence (RRL, Remote Reference Layer) est chargée du système de localisation afin de fournir un moyen aux objets d'obtenir une référence à l'objet distant. Elle est assurée par le package java.rmi.Naming. On l'appelle généralement registre RMI car elle référence les objets. Rmiregistry s'exécute sur chaque machine hébergeant des objets distants.

- **3ème couche : Transport**

La couche de transport permet d'écouter les appels entrants ainsi que d'établir les connexions et le transport des données sur le réseau par l'intermédiaire du protocole TCP. Les packages java.net.Socket et java.net.SocketServer assurent implicitement cette fonction. Les connexions et les transferts de données dans RMI sont effectués par Java sur TCP/IP grâce à un protocole propriétaire JRMP, (Java Remote Method Protocol) sur le port 1099.

A partir de Java 2 version 1.3, les communications entre client et serveur s'effectuent grâce au protocole RMI-IIOP (Internet Inter-Orb Protocol), un protocole normalisé par l'OMG et utilisé dans l'architecture CORBA.Généralement lors d'un appel d'une méthode,

le Stub transmet l'appel à la couche RRL qui assure la connexion avec le serveur en point à point (unicast).

Cette couche transmet la requête à la couche transport qui utilise JRMP au dessus de TCP/IP et transfère la requête en remontant vers le skeleton qui appelle la méthode sur l'objet distant.

Avec Java 2, le skeleton est devenu obsolète (dépassé), une même classe skeleton générique est partagée par tous les objets distants.

En plus, jusqu'à la version 5.0 du J2SE (2005), il fallait utiliser un compilateur de stub appelé RMIC (Java RMI Compiler) pour générer les stub/skeleton avant tout enregistrement sur le registre RMI. [9]

I.5.7 Principe de fonctionnement RMI

- **Coté Serveur**

A la création de l'objet, un stub et un skeleton (avec un port de communication) sont créés coté serveur.

- L'objet serveur s'enregistre auprès d'un annuaire (rmiregistry) en utilisant la classe Naming (méthode rebind).
- L'annuaire (rmiregistry) enregistre le stub de l'objet
- L'annuaire est prêt à donner des références à l'objet Serveur.
- **Coté Client**

L'objet client fait appel à l'annuaire (rmiregistry) en utilisant la classe Naming pour localiser l'objet serveur (méthode lookup).

- L'annuaire délivre une copie du stub.
- L'objet stub est installé et sa référence est retournée au client.
- Le client effectue l'appel à l'objet serveur par appel à l'objet stub. [8]

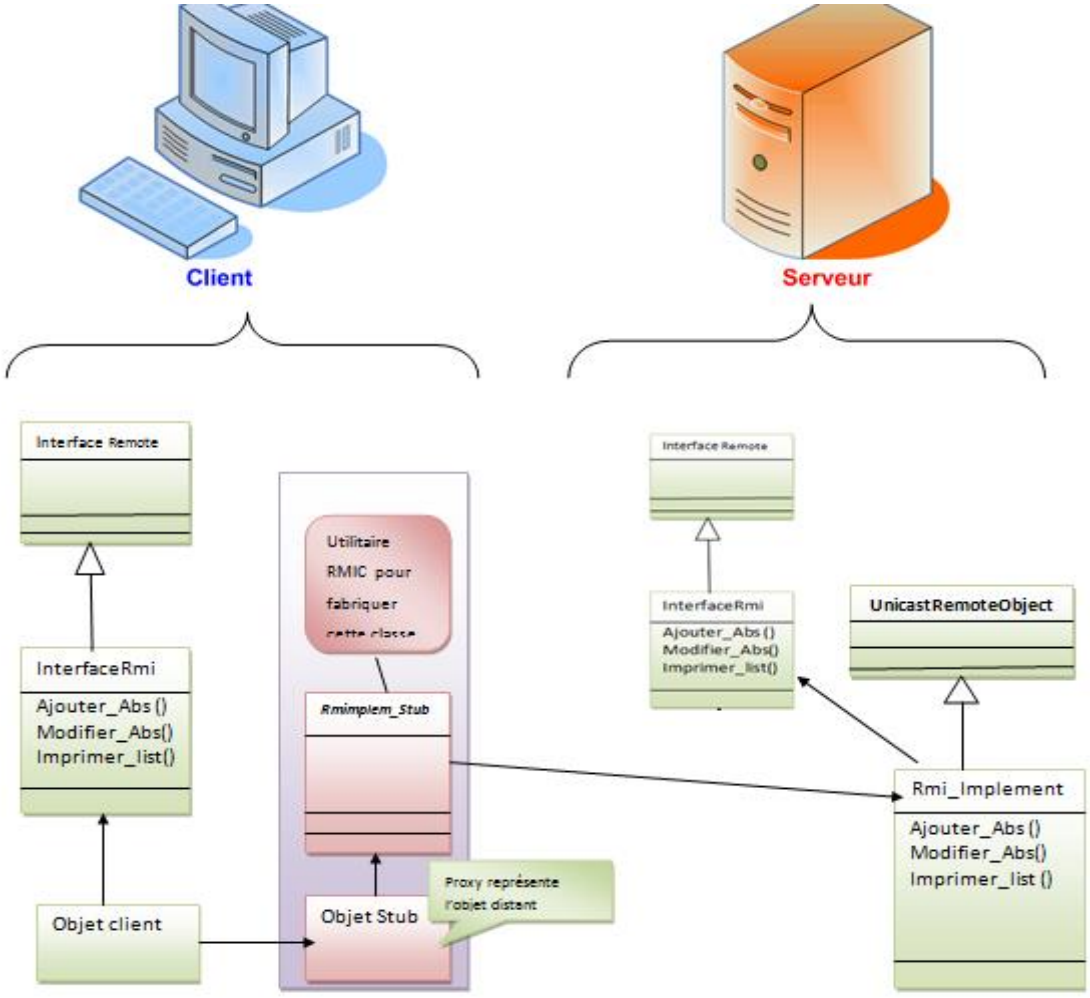


Figure 6 : Principe de fonctionnement RMI

I.5.1 Déploiement de RMI

L'utilisation d'interfaces est un principe important dans RMI, En effet, la définition d'une fonctionnalité et l'implémentation de celle-ci sont deux concepts séparés, Une interface décrit l'interaction entre client et fournisseur du service.

En RMI, on utilise une interface pour coder la définition d'un service et une classe pour coder son implémentation (située à distance), une même interface est implémentée par deux classes : une classe implémente le service (skeleton) sur le serveur, et une classe implémente un proxy (stub), pour le service distant sur le client.

Pour créer une application avec RMI il y a :

- **Coté serveur**
- La définition d'une interface qui contient les méthodes qui peuvent être appelées à distance (l'interface est partagée avec le client), celle-ci doit étendre java.rmi.Remote, et déclarer les méthodes publiques globales de l'objet, c'est-à-dire les méthodes partageables, de plus ces méthodes doivent lancer une exception de type java.rmi.RemoteException.
- L'écriture d'une classe qui implémente cette interface, cette classe doit dériver de java.rmi.server.UnicastRemoteObject.
- L'écriture d'une classe (serveur) qui instanciera l'objet et l'enregistrera en lui affectant un nom dans **RMI Registry.**
- **Côté client**
- L'écriture d'une classe (client) capables d'accéder aux méthodes d'un objet sur le serveur grâce à la méthode Naming.lookup (nom_de_l'objet).
- L'obtention d'une référence sur l'objet distant à partir de son nom.
- L'appel à la méthode à partir de cette référence [9]

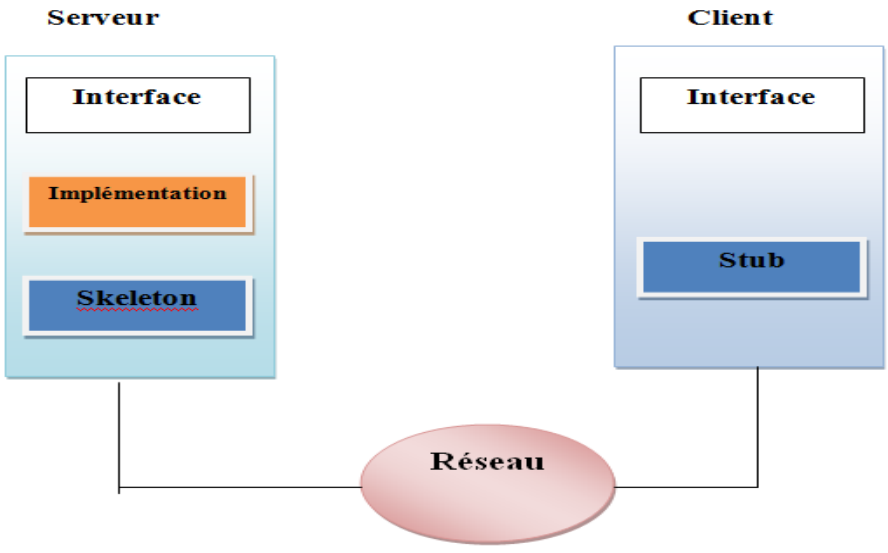


Figure 7 : Déploiement de RMI

I.6 Conclusion

Nous avons abordé dans ce premier chapitre les différents types de l'architecture client/serveur .ainsi nous avons expliqué les principes de l'application RMI qui permet la communication entre machines virtuelles Java sur des machines distantes. En peut l'utiliser pour développer des applications client/serveur basées sur le RMI.

Lors de chapitre suivant on va attaquer la phase analyse pour identifier les besoins ainsi que on va définir les acteurs de notre Système et leur fonctionnement .

.

Chapitre 02 :
Analyse & Conception

II. Chapitre 2 : Analyse & Conception

II.1 Introduction

L'étape d'analyse et conception est le processus d'examen de l'existant et la définition de la future application dont le but est de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client.

Cette partie est consacrée aux étapes fondamentales pour le développement de notre système de gestion des absences. Nous commencerons par une étude sur l'état de l'existant au niveau de la faculté des sciences. Ensuite on spécifiera les besoins de notre application et enfin pour la conception et la réalisation de notre application nous avons choisis de modéliser avec le formalisme UML (Unified Modeling Language) qui offre une flexibilité marquante qui s'exprime par l'utilisation des diagrammes.

II.2 Etude de l'existant

A fin de réaliser un meilleur recueil d'informations sur la situation actuelle de la gestion des absences de notre faculté, nous avons élaboré un questionnaire (voir annexe …). Ce dernier a été rempli par des enseignants ainsi que des secrétaires des quatre départements de notre faculté, que sont Informatique, Mathématique, Physique et Chimie.

L'objectif derrière ce questionnaire est :

- D'analyser le system de gestion des absences actuel.
- D'analyser les problèmes rencontrés avec cette méthode de gestion.
- Déterminer les besoins.
- Acquérir des informations sur les procédures de justification des absences.
- Voici quelques unes des questions que nous avons posées :
- Au niveau de votre département existe-il une application pour la gestion des absences des étudiants ?
- Comment gérez-vous les absences ?
- Quel sont Les problèmes que vous rencontrez avec votre méthode de gestion d'absences ?
- Avez-vous déjà exclu un étudiant à cause des absences ?
- Pensez-vous qu'il serait utile d'informatiser la gestion des absences ?

Les enseignants étaient très positifs envers l'idée d'informatiser la gestion des absences, le but n'étant pas uniquement de pénaliser les étudiants qui s'absentent beaucoup mais aussi du

point de vue gestion, car cela facilitera énormément la sauvegarde des justificatifs d'absences, le suivi des étudiants, même ceux qui ne s'absentent pas, surtout que notre application permet d'alerter son utilisateur dès que le nombre d'absences commence à augmenter, cet utilisateur qu'il soit enseignant ou autre pourra à son tour alerter les étudiants concernés à fin qu'ils prennent leurs précautions.

D'après les réponses que nous avons pu récoltées, nous pouvons conclure que :

- Il n'existe aucune application de gestion d'absences au niveau de notre faculté.
- les absences sont gérées manuellement, et c'est l'enseignant qui s'en occupe.
- Dans la majorité des cas la directive concernant l'exclusion des étudiants qui ont cinq absences justifiées ou 3 non justifiées n'est pas respectée.
- Les retards cumulés ne sont pas pris en considération.

II.3 Problématiques

Les enseignant connait une grande difficulté dans la gestion des absences de leur étudiants, par ce que l'enseignant fait un appel pour chaque séance et il rempli la liste de présence manuellement .Cela engendre des pertes d'informations, et ainsi un retard dans la gestion.

Un enseignant a sa charge plusieurs module qui vient a dire qu'il doit gérer plusieurs étudiants notre objectifs est de lui facilite le suivi de l'assiduité de leur étudiants.

Ci-dessous un ensemble de ces problèmes :

- Risque de perte des listes de présence et cela à cause de l'enregistrement désordonné des fichiers Excel.
- Recherche difficile des absences et ce à cause de l'enregistrement des dossiers sur différents ordinateurs des enseignants.
- Absence de vision globale de l'ensemble des absences enregistrés car il est impossible d'afficher une liste complète de tous les absences d'un étudiant.
- Pas de lien entre les fichiers qui correspondent aux mêmes groupes.
- Perte de temps dans la mise en page des documents, le remplissage des données et la recherche.

II.4 Spécification des besoins

C'est une étape primordiale au début de chaque démarche de développement. Son but est de veiller à développer un logiciel adéquat, sa finalité est la description générale des fonctionnalités du système, en répondant à la question : Quelles sont les fonctions du système?

II.4.1 Besoins fonctionnels

Les services proposés par notre application se résument dans les lignes suivantes :

- **Cote serveur**
- **Accès contrôlé des utilisateurs :** les administrateurs devront s'authentifier avec un nom d'utilisateur et mot de passe propres à chaque administrateur.
- **Création d'un compte utilisateur :** création d'un nouveau client de l'application (enseignant, chef de département, secrétaire…) en remplissant obligatoirement ses informations personnelles afin de pouvoir le créer et l'enregistrer, l'admininistrateur donne a chaque agent un pseudo et un mot de passe pour garantir l'aspect de sécurité.
- **La gestion des étudiants :** ajouter un étudiant en remplissent ses information personnelles, le modifier ou le supprimer.
- **La gestion des filières :** création, modification et la suppression.
- **La gestion des options :** création, modification et la suppression.
- **La gestion des groupes :** création, modification et la suppression.
- **La gestion des modules :** création, modification et la suppression.
- **Coté client**
- **Accès contrôlé des Agents** : les agents devront s'authentifier avec un nom d'utilisateur et mot de passe propres à chaque agent.
- **Création d'un liste de présence:** pour chaque séance une liste de présence est remplit et sauvegarder dans la base de données.
- **Insertion des justifications:** on cas ou un étudiant justifier son absence cette dernière est sauvegarder dans la base de donnée sous format image.

- **Consultation des justificatifs:** l'agent peut afficher les justificatifs.
- **Recherche des absences:** l'agent pourra chercher un étudiant par son nom et son prénom et de là la possibilité d'afficher son absences.
- **impression**: l'agent a la possibilité d'imprimer tous information concernant les absences en format PDF.

Notre système doit répondre aux exigences suivantes :

- Le système doit pouvoir récupérer des informations saisi au niveau de client de chaque entité pour mettre à jour la base des données de l'application.
- L'insertion des absences des étudiants par date, module et séances
- Modification des informations à propos de l'absence.
- Insertion d'une justification d'un étudiants format d'image et la sauvegarder dons la base de donnée.
- Permettre les recherches des absences
- Alerter les étudiants qui ont plus des absences par l'renvoi des emails
- Afficher et imprimer les listes des étudiants exclus.

II.4.2 Besoins non fonctionnels

Les besoins non fonctionnels décrivent toutes les contraintes auxquelles est soumissystème pour sa réalisation et son bon fonctionnement.

- **La sécurité:** nous devons prendre en considération la confidentialité des données, pour cela nous devons restreindre l'accès à ces informations à l'administrateur et aux utilisateurs concernés.
- **La fiabilité et la rapidité:** notre système doit garantir la rapidité et la fiabilité de la recherche des informations, ainsi qu'une gestion optimale des ressources.
- **L'ergonomie:** l'application offre une interface conviviale et facile à utiliser.
- **Une solution ouverte et évoluée :** le code doit être clair pour permettre des futures évolutions ou améliorations.
- **Robustesse et maintenabilité :** l'application doit permettre le stockage des informations et les différents traitements utiles pour le fonctionnement correct, ainsi qu'assurer une gestion exhaustive des erreurs.

II.5 Conception du système

Dans cette partie nous allons introduire les concepts UML fondamentaux pour la spécification des exigences et plus précisément nous allons utiliser le processus unifié qui permettra de modéliser d'une manière claire et précise la structure et le comportement du système.

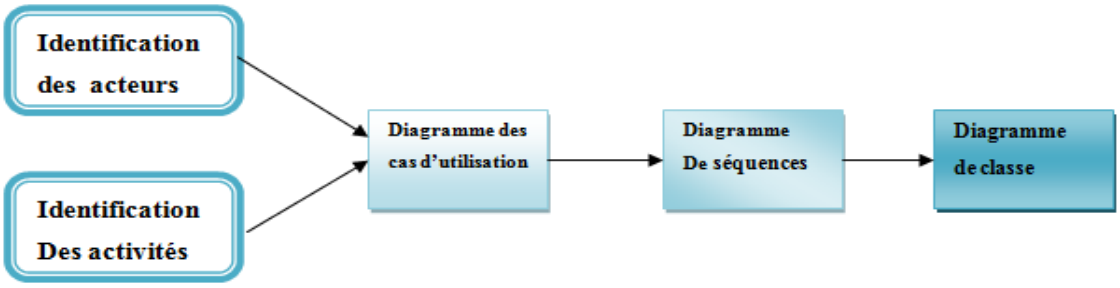


Figure8 : Méthodologies de modélisation

II.6 Le langage de modalisation UML

UML est un langage graphique de modélisation des données et des traitements, fondé sur des concepts orientés objets « langage de modélisation objet unifié ». UML n'est pas une méthode, il convient pour toutes les méthodes objet. UML est en évolution continue.UML propose de décrire un système à l'aide de 13 diagrammes : [12]

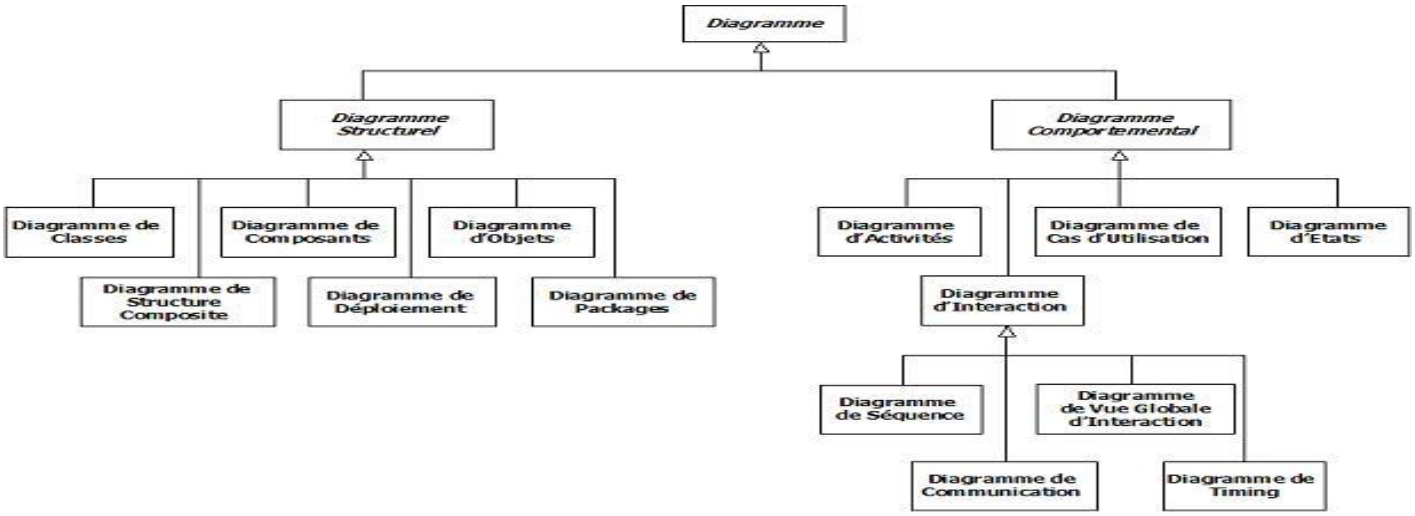


Figure 9 : Classification Des Diagramme UMl

II.7 ArgoUML

Est un outil d'aide à la conception orientée objet.

- **Une application multiplate forme**

ArgoUML est entièrement codé en Java 1.2 et utilise les classes de base de Java (Java Fou ndation Classes). Ceci permet à ArgoUML de fonctionner sur pratiquement n'importe quelle plateforme munie d'une machine virtuelle Javz

ArgoUML est conforme avec la norme UML 1.3 définie p ar l'OMG. Le code pour la représentation interne d'un modèle UML est complètement produit suivant les spécifications de l'OMG. Pour se faire, une bibliothèque spéciale de meta model (NSUML) a été développée par la société Novosofts sous licence GPL. Ceci rend ArgoUML extrêmement flexible pour s'ajuster aux nouvelles normes UML avenir. Cependant quelques caractéristiques avancées d'UML ne sont pas encore disponibles dans les diagrammes. Notamment, il n'existe pas encore de diagramme de séquence.[13]

II.8 Identification des acteurs

Un acteur représente un rôle joué par une entité (utilisateur humain, dispositif

matériel,..), qui interagit directement avec le système étudié. Les acteurs humains pour notre système sont :

Administrateur : rôle des employés qui ont comme charge le bon fonctionnement et la maintenance du système.

Agent : tous personne intéressé par la gestion des absences enseignant , chef de département, secrétaire…etc.

II.9 Identification des activités

Pour bien encadrer notre du système nous avons définie un ensemble d'activités pour chaque acteur qui serons développées par la suite :

- Activité de l'administrateur
- Gestion des agents
- Gestion des étudiants
- Gestion des filières, options, groupes.
- Impression
- Gestion des la base de donnée.
- Maintenir de système.
- Activité de l'agent
- Authentification au système.

- Insertion des absences.
- Insertion des justifications.
- Recherche des absences.
- Afficher les étudiants exclus.
- Impression des informations.

II.10 Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation est un graphe d'acteurs, un ensemble de cas d'utilisation englobés par la limite du système, des associations de communication entre les acteurs et les cas d'utilisation, et des généralisations entre cas d'utilisation.

Il est destiné à représenter les besoins des utilisateurs par rapport au système.

II.10.1 Identification des acteurs

Les acteurs d'un système sont les entités externes à ce système qui interagissent avec lui. Dans notre application, le seul acteur qui interagit avec le système est l'agent de saisie du bureau des entrées. [14]

II.10.2 Identification des cas d'utilisations

Un cas d'utilisation est utilisé pour définir le comportement d'un système ou la sémantique de toute autre entité sans révéler sa structure interne. Chaque cas d'utilisation spécifie une séquence d'action, y compris des variantes, que l'entité réalise, en interagissant avec les acteurs de l'entité. La responsabilité d'un cas d'utilisation est de spécifier un ensemble d'instances, où une instance de cas d'utilisation représente une séquence d'actions que le système réalise et qui fournit un résultat observable par l'acteur. [14]

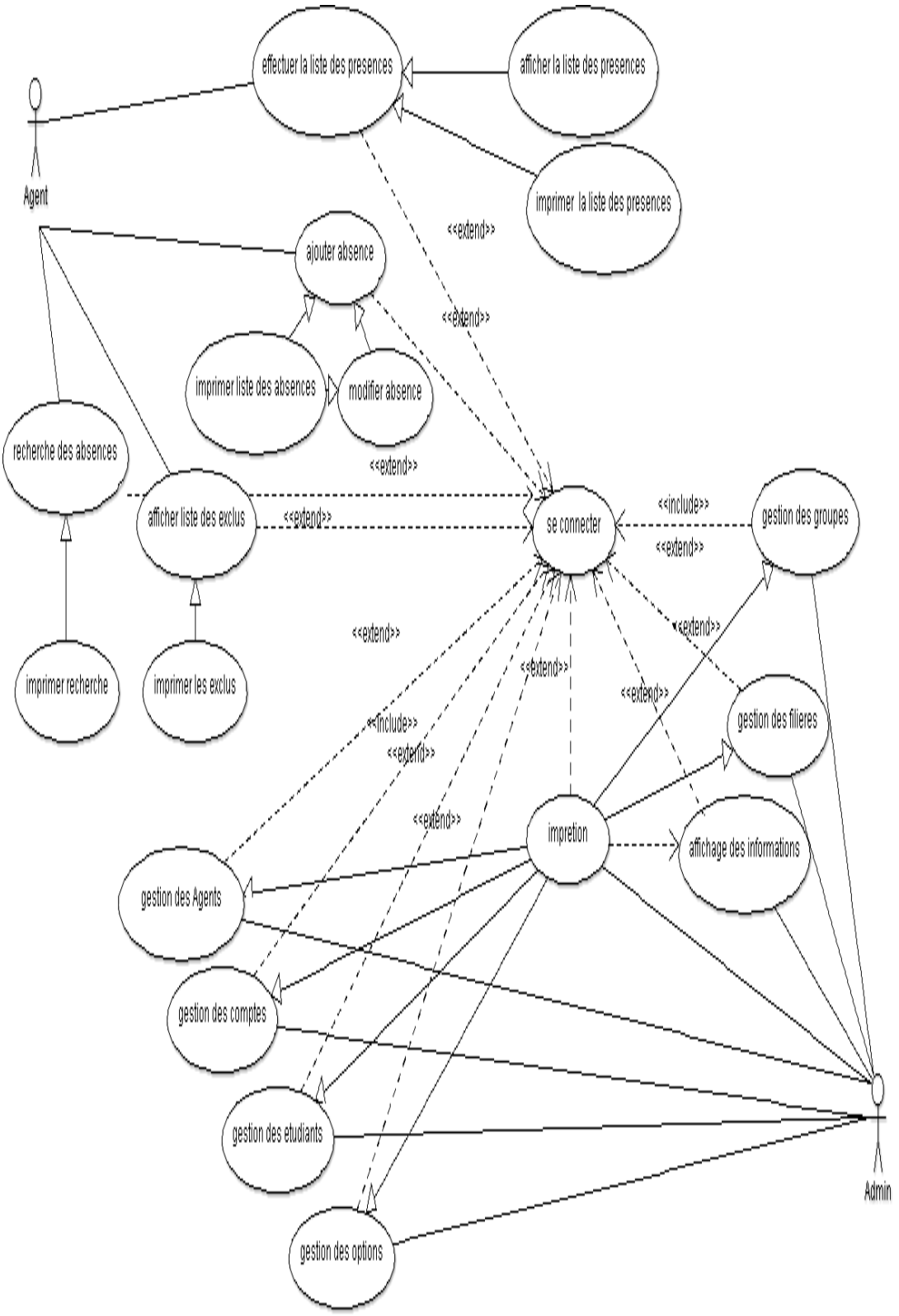


Figure 10 : Diagramme de cas d'utilisation

II.11 Spécification des tâches et scenarios du système

Les scenarios décrivant chacune des tâches définies auparavant sont récapitulés dans le tableau suivant :

Acteur	*tâches*	Scénario
Administrateur	T1: Se connecter	S1 : Saisir le nom d'utilisateur et le mot de passe
		S2: Connexion
	T2 : Accéder a l'application (Gérer la base de Données)	S3: gérer les utilisateurs
		S4 : gérer les clients (Agents)
		S5 : gérer les étudiants
		S6 : gérer les filières, options, niveaux, groupes
		S7: Consulter les listes des étudiants
		S8: imprimer
		S9: afficher
	T3 : Se déconnecter	S10: Déconnexion
Client (Agent)	T4 : Se connecter	S11: Saisir le nom d'utilisateur et le mot de passe S12 : Connecter
	T5: Accéder a l'application	S13: Consulter les listes des étudiants
		S14 : Créer la liste de présence
		S15 : Modifier les absences
		S16 : afficher liste des étudiants exclu par module
		S17 : Imprimer les listes des étudiants
	T8 : Se déconnecter	S18: Déconnexion

Tableau 1 : Spécification des tâches et scenarios du système

II.12 Diagramme de séquence

Il permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets. Un diagramme de séquence montre une interaction présentée en séquence dans le temps. En particulier, il montre aussi les objets qui participent à l'interaction par leur "ligne de vie" et les messages qu'ils échangent présentés en séquence dans le temps.

Voici quelques notions de base du diagramme :

Scénario : une liste d'actions qui décrivent une interaction entre un acteur et le système.

Interaction : un comportement qui comprend un ensemble de messages échangés par un ensemble d'objets dans un certain contexte pour accomplir une certaine tâche.

Message : Un message représente une communication unidirectionnelle entre objets qui transporte de l'information avec l'intention de déclencher une réaction chez le récepteur. [14]

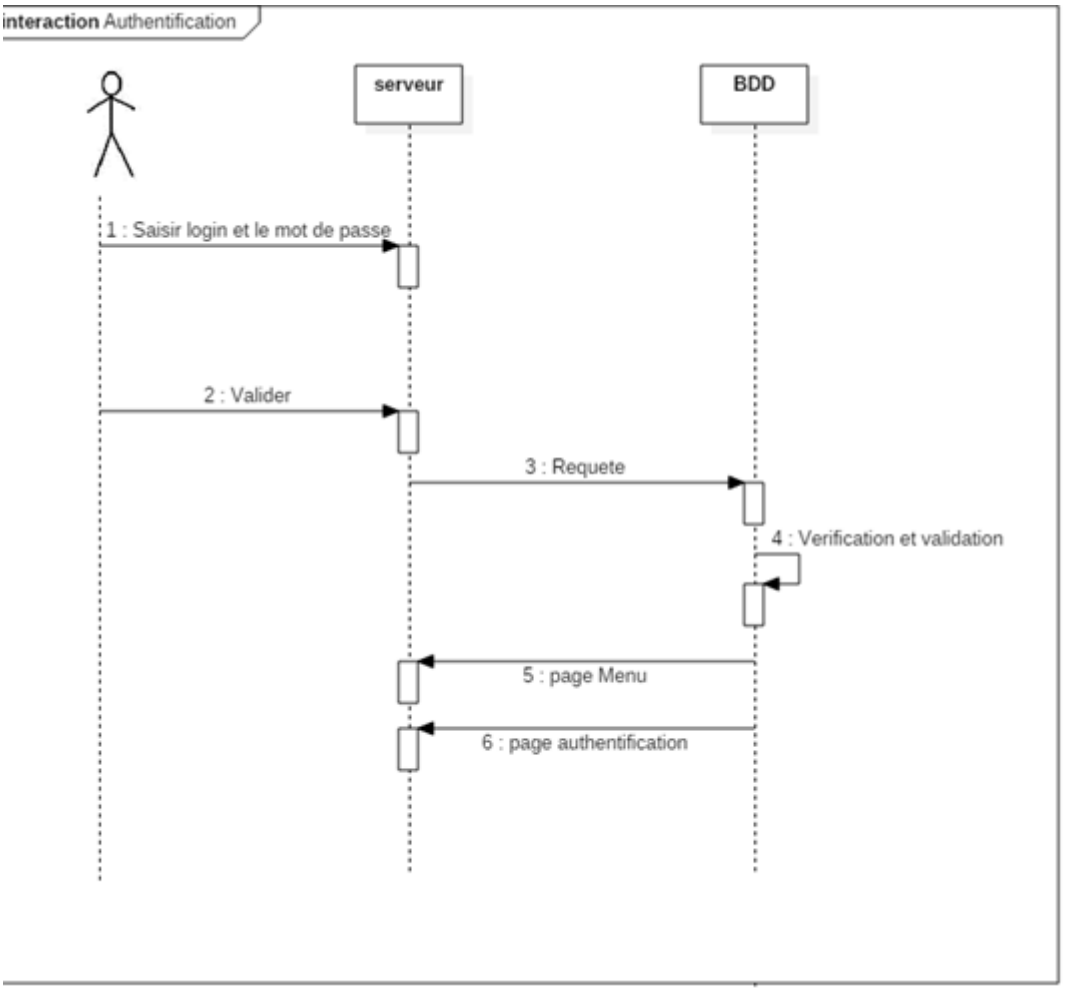


Figure 11 : Diagramme de séquence (exemple d'authentification)

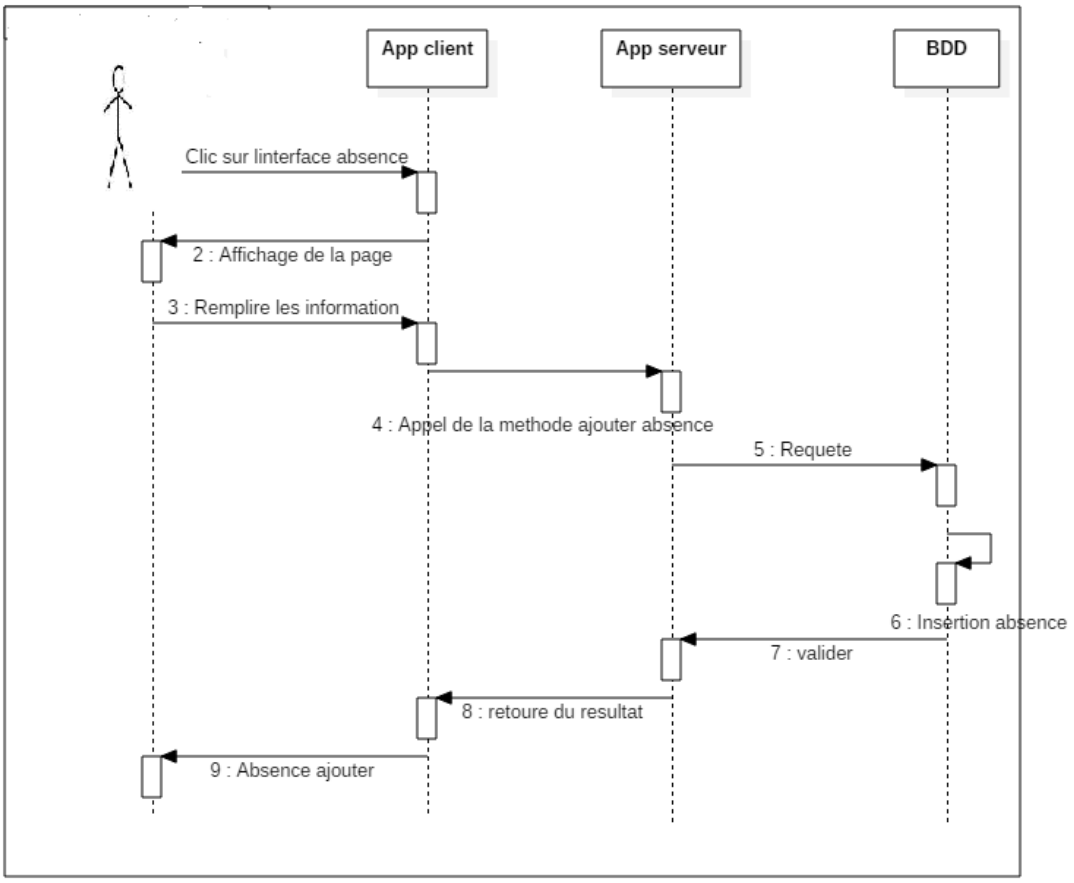


Figure 12 : Diagramme de séquence (exemple d'ajouter absence)

II.13 Diagramme de classes

C'est une collection d'éléments de modèle statique, tels que des classes, des interfaces et leurs relations, connectés entre eux comme un graphe.

Il représente la description statique du système en intégrant dans chaque classe la partie dédiée aux données et celle consacrée aux traitements. C'est le diagramme pivot de l'ensemble de la modélisation d'un système.

Identification des classes

Une classe est une description d'un groupe d'objets partageant un ensemble commun de propriétés (les attributs), de comportements (les opérations) et de relations avec d'autres objets (les associations et les agrégations).

Une classe contient :

- Des attributs (ou champs, ou variables d'instances) : Les attributs d'une classe décrivent la structure de ses instances (les objets).
- Des méthodes (ou opérations de la classe) : Les méthodes décrivent les opérations qui sont applicables aux instances de la classe.

Une agrégation est une association correspondant à une relation qui lorsqu'elle est lue dans un sens signifie "est une partie de" et lorsqu'elle est lue dans l'autre sens elle signifie "est composé de".[14]

A partir des diagrammes de séquence et de la liste d'objets tirés, nous pouvons dresser le diagramme de classe qui expose les relations entre ces objets.

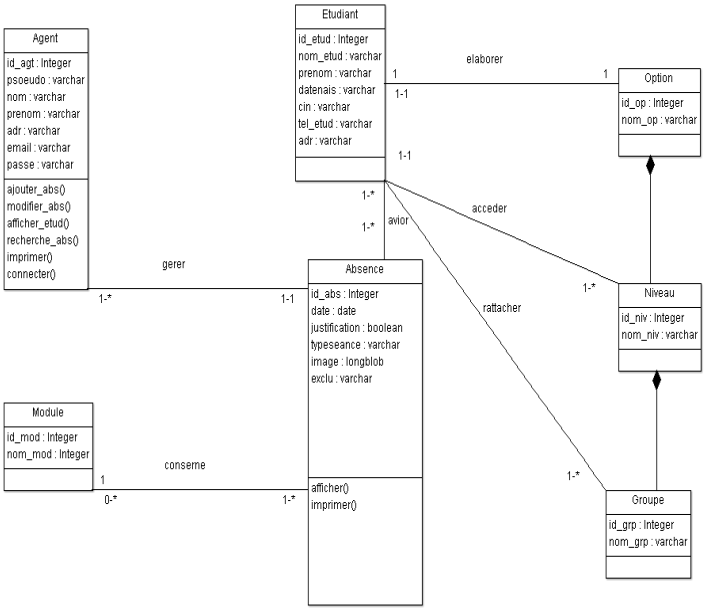


Figure 13 : Diagramme de Classe

II.1 Conclusion

Dans ce chapitre, nous nous sommes concentrés sur les aspects analytique et conceptuel de notre application ainsi que la base de données qui interagit avec elle.

Pour la phase analyse, nous avons défini les différents cas d'utilisations puis, nous les avons traduits en diagrammes UML. Tel que le diagramme de classe ,diagramme de cas d'utilisation et le diagramme de séquence. La phase d'analyse et conception est une étape très importante pour la bonne réalisation du projet.

La phase suivante est la mise en œuvre de notre application. C'est dans le chapitre suivant que sera décrite cette dernière ainsi que Les outils de développement et on va détailler le fonctionnement global de notre application.

III. Chapitre 03 Implémentation

III.1 Introduction

Il nous reste à mettre les concepts de RMI en pratique sur une application clients/serveur de gestion d'absence pour cela nous avons choisis le langage JAVA avec l'environnement ECLIPSE , et nous avons utilisés ``Swing`` pour développer les interfaces *graphique, ainsi MySQL pour le but de crée et interroger les tables de la base de donnée , on* utilise EasyPHP pour l'administration .

Ce chapitre couvre la création et la mise en œuvre des différents programmes, interfaces et bases de données, qui constituent notre application. Nous décrivons l'environnement de création du système, ensuite nous présentons quelques interfaces résultantes.

III.2 Java

Java intègre les concepts les plus intéressants des technologies informatiques récentes dans une plate-forme de développement riche et homogène,Java a été développée par SUN au début des années 90 dans une optique de plus grandportabilité d'une machine à une autre, et d'une grande fiabilité, il s'agit donc d'un environnement de Programmation orientée objet . Il est possible d'utiliser Java pour créer des logiciels dans des environnements très diversifiés :

- Applications sur client lourd (JFC) ;
- Applications Web, côté serveur (servlets,JSP,Struts,JSF) ;
- Applications réparties (EJB) ;
- Applications sur carte à puce (JavaCard).

Ces applications contiennent des nombreuses fonctionnalités :

- Accès à des bases de données (JDBC) ;
- Traitement XML ;
- Web services (JAXM) ;
- Accès à des annuaires (JNDI).

Historique

En raison des difficultés rencontrées avec C++, il était préférable de créer un nouveau langage autour d'une nouvelle plate-forme de développement. Deux développeurs de chez SUN, James Gosling et Patrick Naughton se sont attelés a cette tâche afin d'avoir une plateforme et un langage idéal pour le développement d'applications sécurisées, distribuées, robuste et portable sur de nombreux périphériques et systèmes embarqués interconnectés en réseau mais également sur Internet (clients légers) et sur des stations de travail (clients lourds), n'oubliant pas la richesse de ses API fournis en standard ou par des tiers commerciaux

D'abord surnommé C++-- (C++ sans ses défauts) puis OAK, mais il s'agissait d'un nom déjà utilisé dans le domaine informatique, il fut finalement baptisé Java mot d'argot voulons dire café en raison des quantités de café ingurgité par les programmeurs et notamment par ses concepteurs. [15]

Caractéristique de java

- **La programmation Orientée Objet**

Il est impossible de parler de Programmation Orientée Objet sans parler d'*objet*, bien entendu. Tâchons donc de donner une définition aussi complète que possible d'un *objet*.

Un objet est avant tout une structure de données (Objet =Données+Méthodes). Autrement, il s'agit d'une entité chargée de gérer des données, de les classer, et de les stocker sous une certaine forme. En cela, rien ne distingue un *objet* d'une quelconque autre structure de données. La principale différence vient du fait que l'*objet* regroupe les données et les moyens de traitement de ces données.

Un objet rassemble de fait deux éléments de la programmation procédurale.

- **Les champs**

Les champs sont à l'objet ce que les variables sont à un programme : ce sont eux qui ont en charge les données à gérer. Tout comme n'importe quelle autre variable, un *champ* peut posséder un type quelconque défini au préalable : nombre, caractère... ou même un type objet.

- **Les méthodes**

Les méthodes sont les éléments d'un objet qui servent d'interface entre les données et le programme. Sous ce nom obscur se cachent simplement des procédures ou fonctions destinées à traiter les données.

Les champs et les méthodes d'un objet sont ses **membres**.

Si nous résumons, un objet est donc un type servant à stocker des données dans des champs et à les gérer au travers des méthodes.

- **Encapsulation des données**

L'accès aux données des objets est réglementé car les données privées sont accessibles uniquement par les fonctions membres. Les données publiques sont accessibles directement par l'instance de l'objet par conséquence un objet n'est vu que par ses spécifications. Une modification interne est sans effet pour le fonctionnement général du programme et une meilleure réutilisation de l'objet.

- **L'Héritage**

Permet de définir les bases d'un nouvel objet à partir d'un objet existant. Aussi le nouvel objet hérite des propriétés de l'ancêtre et peut recevoir de nouvelles fonctionnalités. Enfin la possibilité de la réutilisation de l'objet. [16]

III.3 Eclipse

La nouvelle version de l'IDE, apporte le support natif de Java 8.

L'environnement de développement open source Eclipse évolue pour répondre aux besoins des développeurs. Eclipse 4.4 est disponible en téléchargement. Baptisé Luna (déesse de la Lune dans la mythologie romaine).Eclipse Luna se compose de 76 projets, pour un total d'environ 10 millions de nouvelles lignes de code. 339 contributeurs à travers le monde ont participé à cet effort. Huit projets ont rejoint le « simulatenous release train » : EMF Client Platform, EMF Store, Sirius, BPMN2 et BPMN2 Modeler Project, Paho QVTd et XWT. Sans être exhaustifs, les principales nouveautés autres que le support de Java 8 sont :

- Le projet Sirius, qui permet de construire de manière simple et totalement nouvelle des éditeurs, basé sur le framework de méta-modélisation EMF et sa couche graphique spécifique GMF. Avec Eclipse Sirus, le développement d'éditeurs devient déclaratif ;
- Un nouveau thème graphique "sombre" très proche de ce qui a été proposé par les dernières versions de l'environnement de développement IntelliJ ;
- A possibilité de créer des vues de type "E4" depuis un plugin. La nouvelle API E4 ne sera donc plus réservée aux applications Eclipse RCP ;
- Une amélioration sensible du Workbench, qui permet notamment de pouvoir séparer un même éditeur en 2 parties, soit horizontalement, soit verticalement ;
- L'éditeur graphique d'EMF (Editor Diagram) a été entièrement rebâti en se basant sur le projet Sirus ;
- Une amélioration de la bibliothèque EGIT, l'implémentation Java de GIT. [17]

III.4 JDBC (java database connectivity)

JDBC est une API fournie avec Java (depuis sa version 1.1) permettant de se connecter à des bases de données, c'est-à-dire que JDBC constitue un ensemble de classes permettant de

développer des applications capables de se connecter à des serveurs de bases de données (SGBD).

L'API JDBC a été développée de telle façon à permettre à un programme de se connecter à n'importe quelle base de données en utilisant la même syntaxe, c'est-à-dire que l'API JDBC est indépendante du SGBD.

De plus, JDBC bénéficie des avantages de Java, dont la portabilité du code, ce qui lui vaut en plus d'être indépendant de la base de données d'être indépendant de la plate-forme sur laquelle elle s'exécute. [18]

Principaux éléments de JDBC

- **JDBC Driver API**

L'interface java.sql.Driver est destinée aux développeurs de drivers (pilotes) désirant interfacer un SGBD à Java en utilisant JDBC. La programmation d'un driver JDBC consiste à implanter les éléments définis dans les interfaces abstraites de l'API JDBC.

- **DriverManager**

L'interface java.sql.DriverManager gère la liste des drivers JDBC chargés dans la machine virtuelle Java et la création des connexions TCP. Il sert également à mettre en correspondance les URL utilisés par les connexions avec les drivers à disposition. Ainsi, le DriverManager décortique les URL afin d'en extraire le sous-protocole et le service, puis recherche dans sa liste de drivers celui (ou ceux) capable(s) d'établir la connexion. java.sql.Connection Gère les connexions existantes, crée les requêtes, gère les transactions.

Une fois créée, une Connexion va servir à envoyer des requêtes au SGBD et à récupérer les résultats, ces deux tâches étant effectuées à l'aide des interfaces Statement (requête) et ResultSet (ensemble résultat). [18]

JDBC et SQL

SQL (sigle de Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à effectuer des opérations sur des bases de données relationnelles. La partie langage de manipulation de données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.

Créé en 1974, normalisé depuis 1986, le langage est reconnu par la grande majorité des systèmes de gestion de bases de données relationnelle (abrégé SGBDR) du marché. SQL fait partie de la même famille que les langages SEQUEL (dont il est le descendant), QUEL ou MySQL est un gestionnaire de base de données libre. Il est très utilisé dans les projets libres et dans le milieu industriel. [19]

Requêtes SQL dans JDBC

Les requêtes SQL sont représentées dans JDBC à l'aide des interfaces Statement, PreparedStatement et CallableStatement. Chacune de ces interfaces est spécialisée dans un type particulier de requêtes,

Une fois obtenue auprès d'une Connection, une requête peut être exécutée, son résultat prenant la forme d'un ResultSet, dans le cas d'un SELECT. Ce résultat peut alors être vérifié (valeur NULL, ensemble vide, etc.) et parcouru ligne par ligne. Les ordres DELETE, INSERT, UPDATE quant à eux retournent un entier.

La méthode executeQuery() exécute une requête et retourne le résultat sous forme d'un

ResultSet. Un ResultSet est un ensemble de lignes (chaque ligne représentant un tuple de la relation résultat); chaque ligne comporte le même nombre de colonnes (chaque colonne représentant un attribut de la relation résultat). [19]

III.5 EasyPHP et MySQL

EasyPHP permet d'installer MySQL, une base de données, le troisième et inséparable membre du trio Apache/PHP/MySQL. Une base de données est un programme permettant de gérer une grande quantité de données en les organisant sous forme de tables.

Vous n'avez alors plus à vous occuper de la manière dont les données sont stockées sur le disque dur, de simples instructions permettent d'ajouter, de supprimer, de mettre à jour et surtout de rechercher des données dans une base de données. On peut de plus accéder très facilement à une base de données MySQL à partir de PHP, ce qui permet de développer des sites web très performants et interactifs (par exemple, le forum de Developpez.com). EasyPHP joint PHPMyAdmin à MySQL, un outil écrit en PHP permettant de gérer vos bases de données MySQL. En utilisant EasyPHP, vous pouvez installer un serveur web complet, qui vous permettra de faire tous vos tests de pages PHP en toute facilité.

MySQL dérive directement de SQL (Structured Query Language) qui est un langage de requêtes vers les bases de données relationnelles.

Il en reprend la syntaxe mais n'en a pas toute la puissance. [20]

III.6 Présentation de l'application

Notre projet consiste à réaliser une application de gestion des absences pour notre faculté des sciences cette dernière est de type Client-serveur basée sur Java RMI, dans un réseau locale.

c-à-d informatiser la procédure d'ajouter les absences des étudiants au niveau des séances de chaque module , cette application permet aussi d'ajouter les justification d'absences des étudiants et les afficher et elle offre la possibilité de signaler les étudiants qui ont exclu .

III.7 Développement de gestion d'absences

Pour récapituler la réalisation de notre application , nous allons exposer les étapes de déroulement de notre travail.

Nous allons commencer par le choix des outils. En effet, les outils qui ont permis le développement de notre application ont été choisis pour des soucis de compatibilités, notamment dus à la récence de certaines technologies.

La réalisation du projet s'est déroulée en plusieurs étapes :

Nous avons utilisé le modèle relationnel afin de concevoir le schéma de la base de données.

Un premier projet de test a été développé afin de tester les connexions à la base de données, le bon déploiement du projet sur le serveur d'applications ainsi que les opérations de base comme l'ajout et la suppression. Nous avons créer une JFrame d'authentification pour nous assurer qu'il y avait une bonne communication entre base de donnée et serveur. Après nous être assurés de la bonne connexion entre eux , nous avons pu continuer le développement du projet .

Un deuxième projet de test a été développé afin de tester la connexion client-serveur sur deux machines pour cela nous avons créé un réseau ad-hoc sécurisé par un mot de passe pour que le client et le serveur soient sur le même réseau (voir annexe), nous avant réussi d'établir une connexion client serveur RMI sur deux machine c'est a base de ce teste nous avons pu poursuivre notre travail.

Parmi les services que l'application doit fournir l'affichage des justifications d'absence des étudiants .Aussi il offre le service d'impression cote client et cote serveur. Ainsi que l'envoi d'email d'avertissement au étudiants qui ont exclu ou proche d'être exclu dons un module. De même que notre application permet les opération de base tel que l'ajout, modification et la suppression.

III.8 Application gestion des absences

Les interfaces graphiques de l'application sont très importantes, car elles permettent de faciliter le dialogue entre l'homme et la machine ainsi que d'améliorer les performances de l'application. Dans la conception des interfaces de notre application nous avons respecté un ensemble des choix ergonomiques comme la lisibilité, la compréhension, etc.

- **Cote Serveur**
- Auhtentification
- Deconnexion
- Indication du mot de passe
- Menu administrateur
- Lancer/arret du serveur
- Gestion des utilisateurs
- Gestion des etudiants
- Gestion des ensiegnants
- Gestion des filieres
- Gestion des options
- Gestion des modules
- Consultation des justifications
- **Cote Client**
- Auhtentification
- Deconnexion
- Indication du mot de passe
- Ajouer absence
- Modifier absence
- Recherche des absences
- Exlu et retard
- Consultation des justification

Dans ce qui suit une présentation des captures écrans des plus importantes tâches de l'application.

- **Cote Serveur**

Authentification

Cette première capture présente l'interface d'authentification dans laquelle l'administrateur doit saisir son nom d'utilisateur et son mot de passe pour commencer à utiliser notre application. Cette interface constitue la fenêtre d'accueil de notre application.

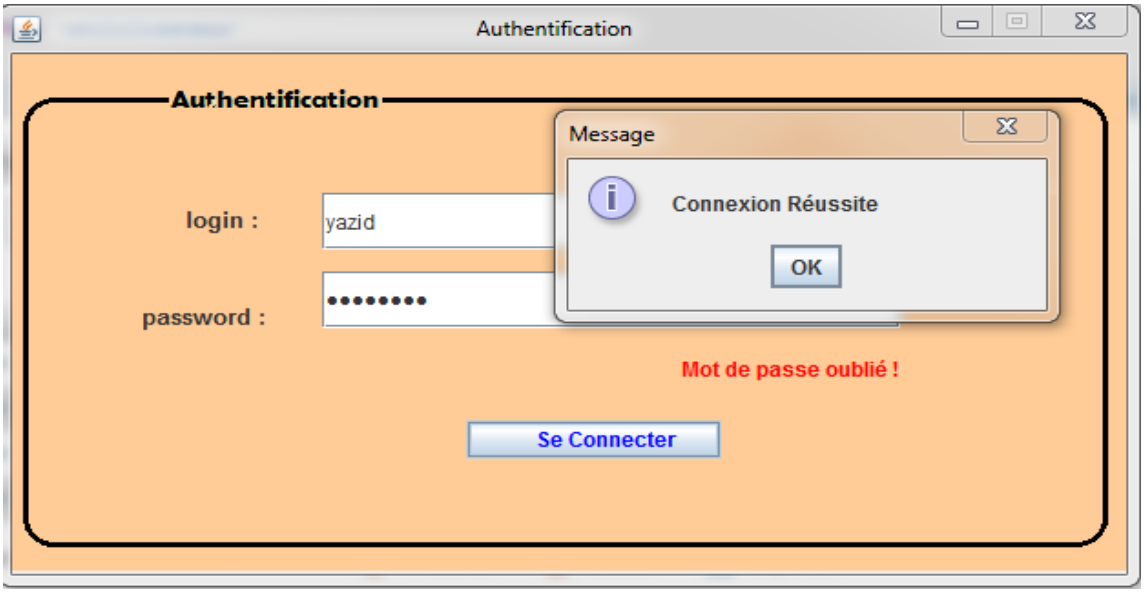


Figure 14 : Fenêtre d'authentification

Cette étape met en valeur l'aspect sécurité : nous vérifions la disponibilité du compte utilisateur et nous lui attribuons les droits et privilèges nécessaires.

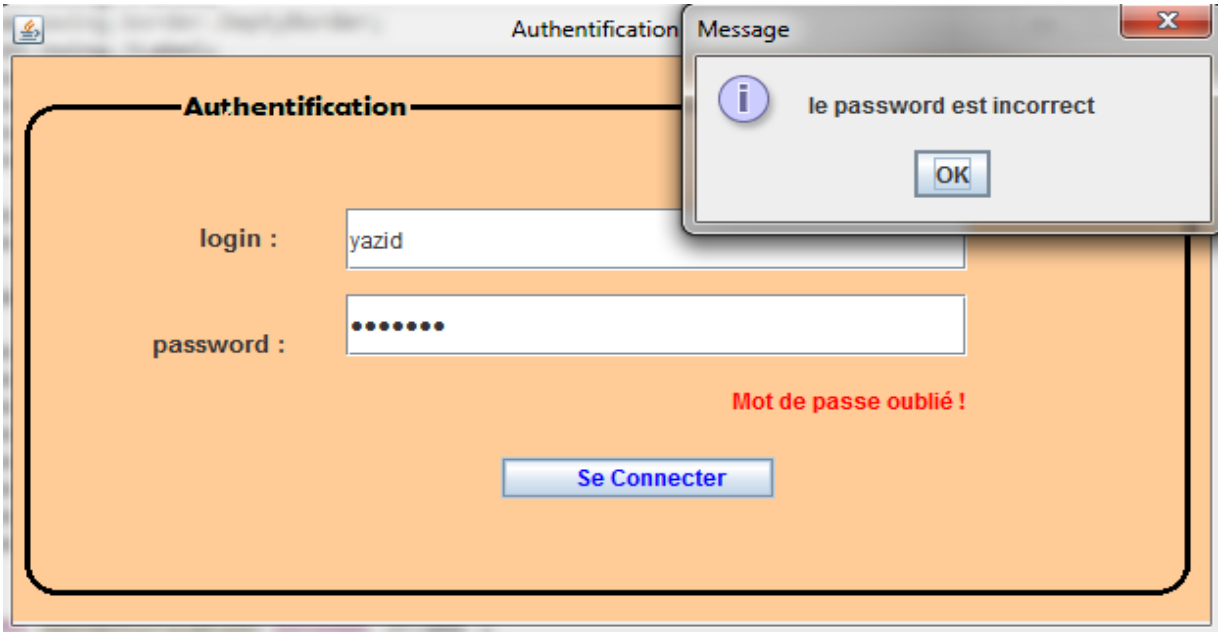


Figure 15 Vérification de mot de passe

On cas ou le mot de passe est oublié l'application offre la possibilité pour le récupérer par une indication du trois premier lettre de mot de passe.

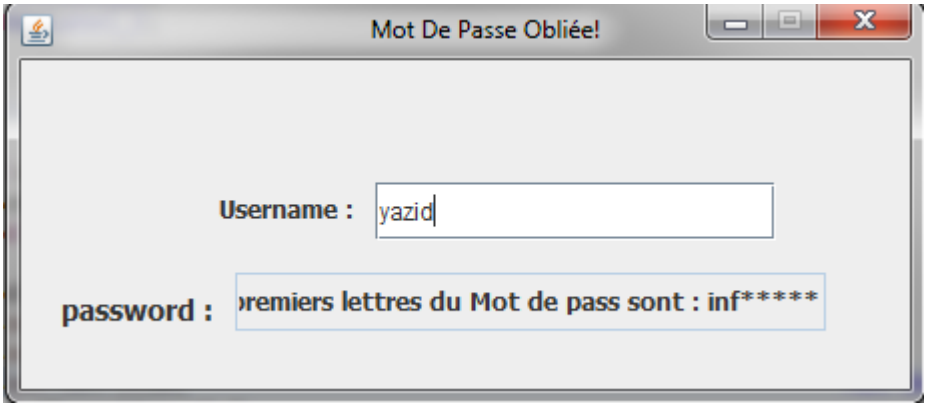


Figure 16 : indication de mot de passe

Menu Principal

Une fois l'administrateur authentifié, le menu principal s'affiche à l'écran.

Sur le menu s'affiche l'heure et la date.

Le serveur est lancé depuis le menu.

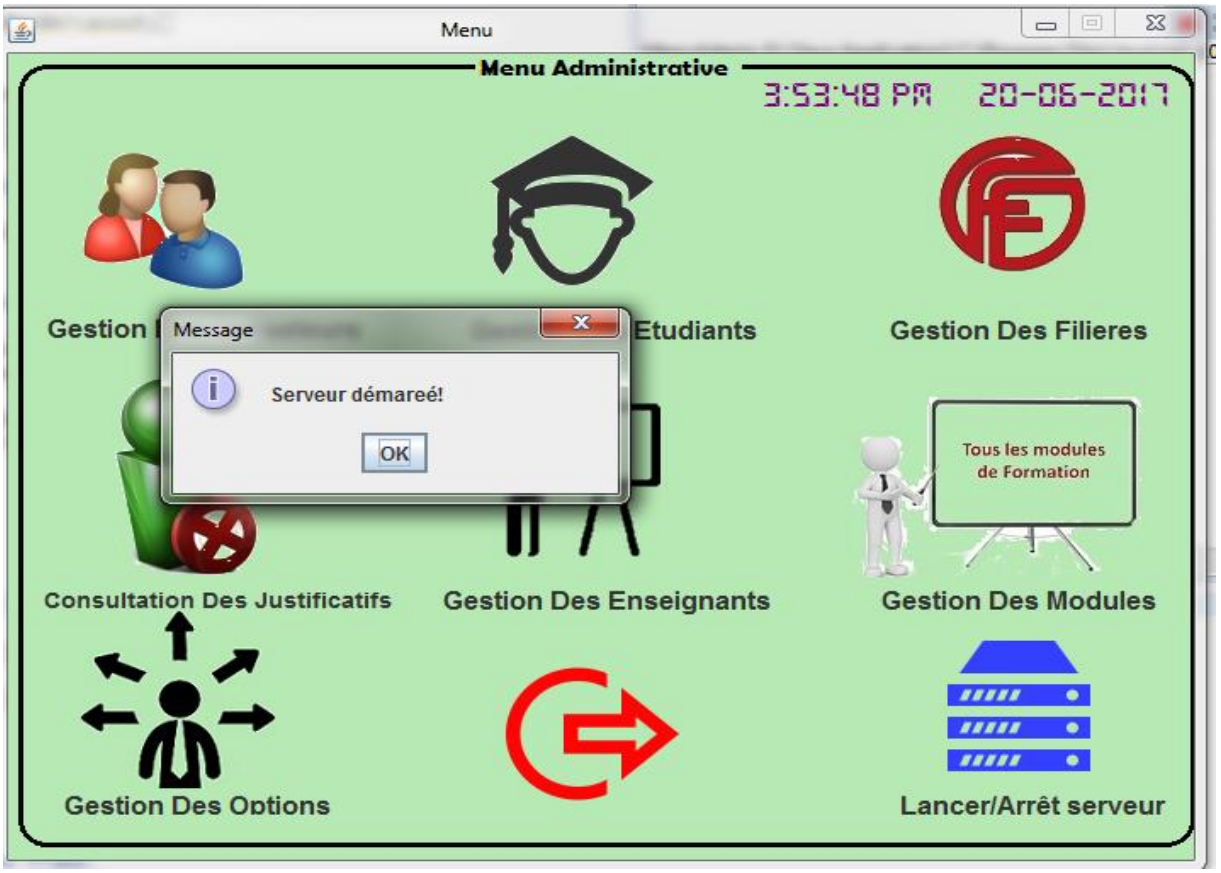


Figure 17 : Menu principal serveur

Gestion des étudiants

Cette interface permet a un administrateur d'ajouter un étudiant de le modifier ou

de supprimer, aussi cette interface offre le service d'impression des liste des étudiants (voire annexe).

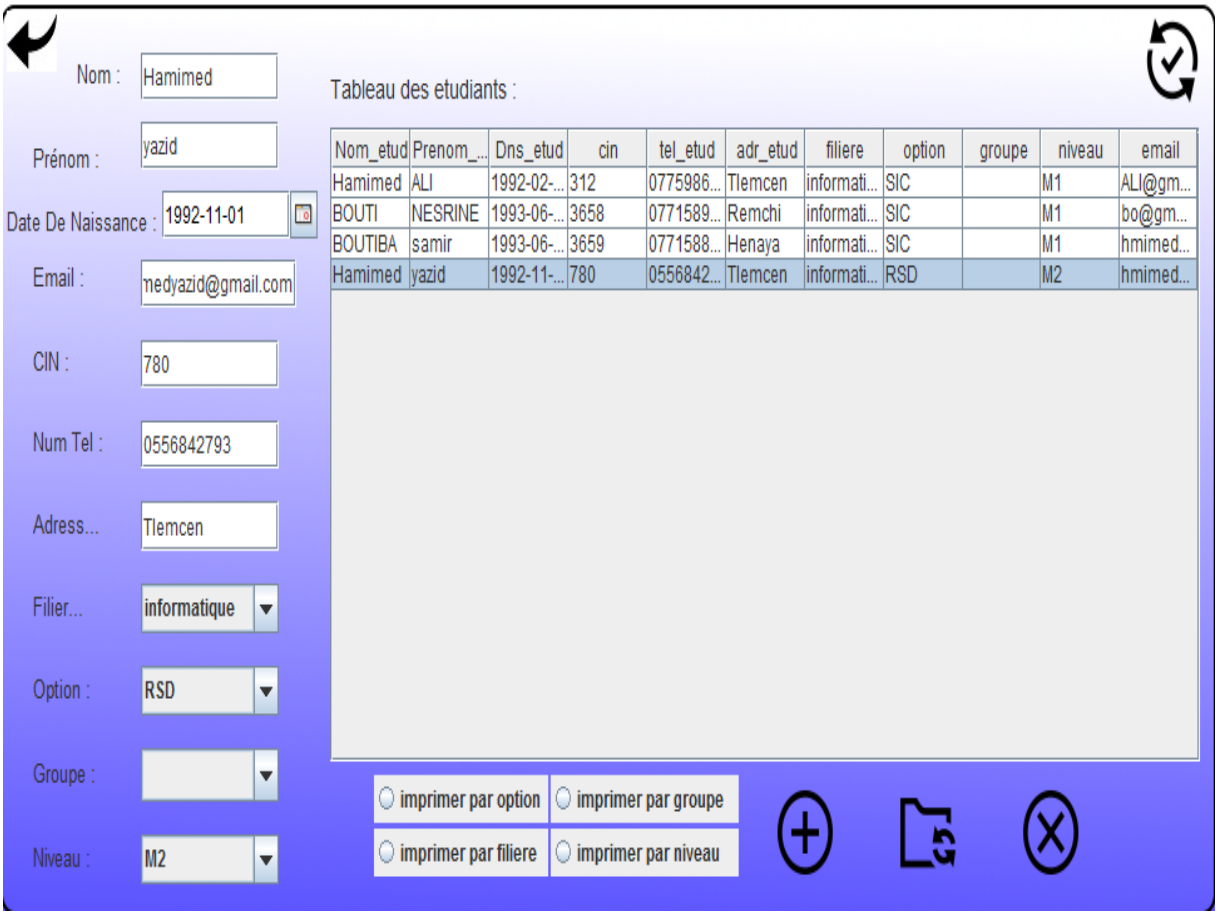


Figure 18 : gestion des étudiants

- **Cote Client**

Authentification

Cette première capture présente l'interface d'authentification dans laquelle le client doit saisir son nom d'utilisateur et son mot de passe pour commencer à utiliser notre application. Cette interface constitue la fenêtre d'accueil de notre application cote client.

Cette étape met en valeur l'aspect sécurité : nous vérifions la disponibilité du compte utilisateur et nous lui attribuons les droits et privilèges nécessaires.

On cas ou le mot de passe est oublié l'application offre la possibilité pour le récupérer par une indication du trois premier lettre de mot de passe.

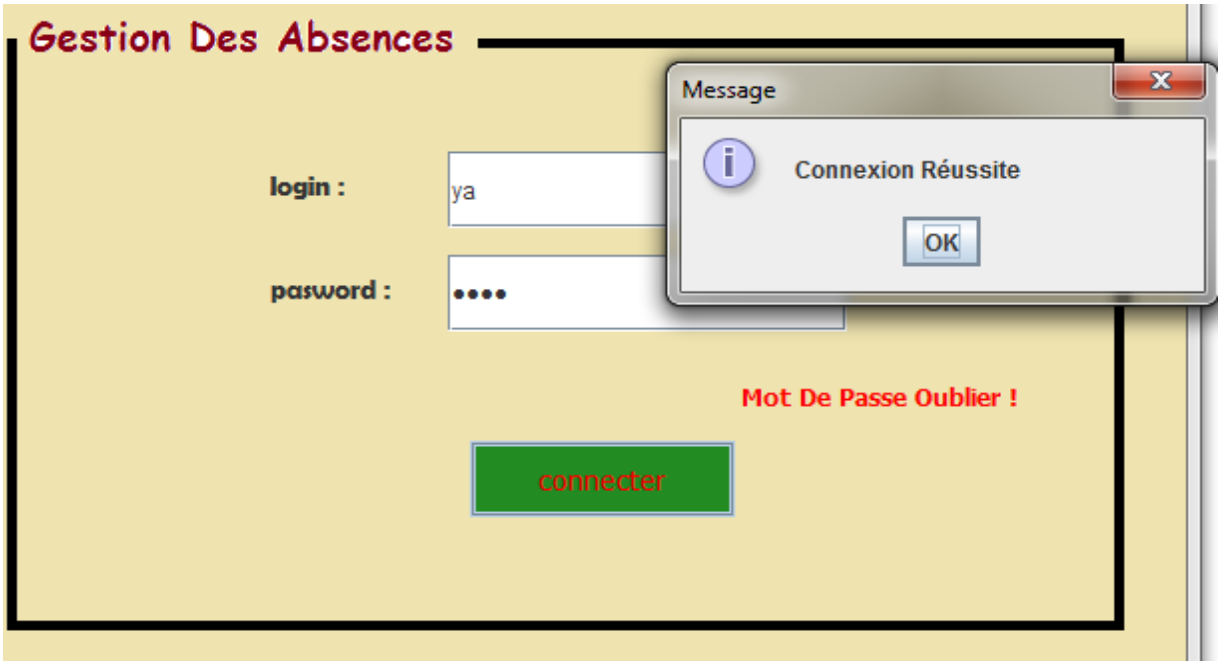


Figure 19 : Fenêtre d'authentification agent

Menu Client

Le menu client est l'interface qu'il peut diriger l'agent à l'ensemble des services offert par notre application

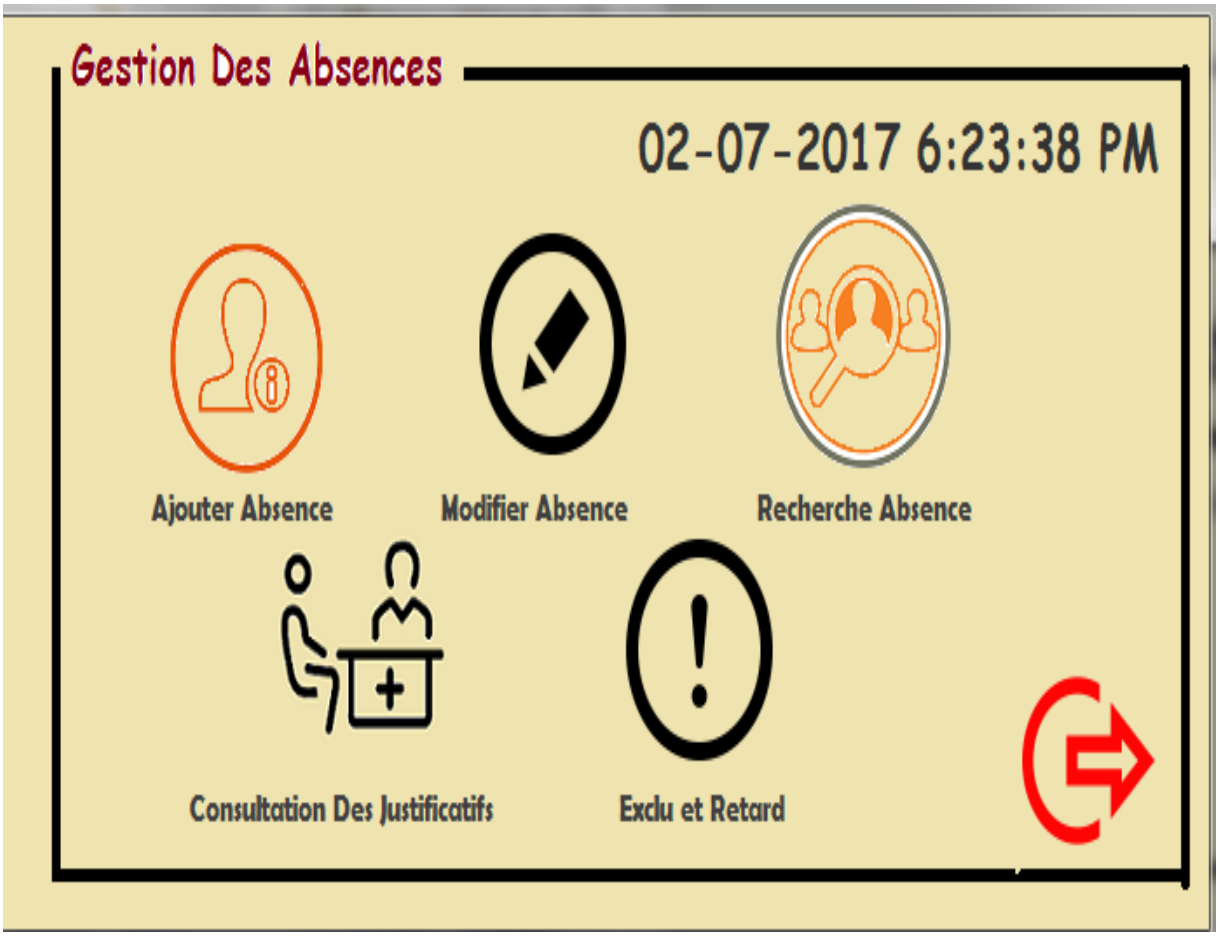


Figure 20 : Le menu cote client

Ajouter Absence

Cette IHM contient la partie insertion de la liste de présence pour chaque séance d'un module dans une date données, les étudiants sont classer par filiere,niveao,option et groupe.

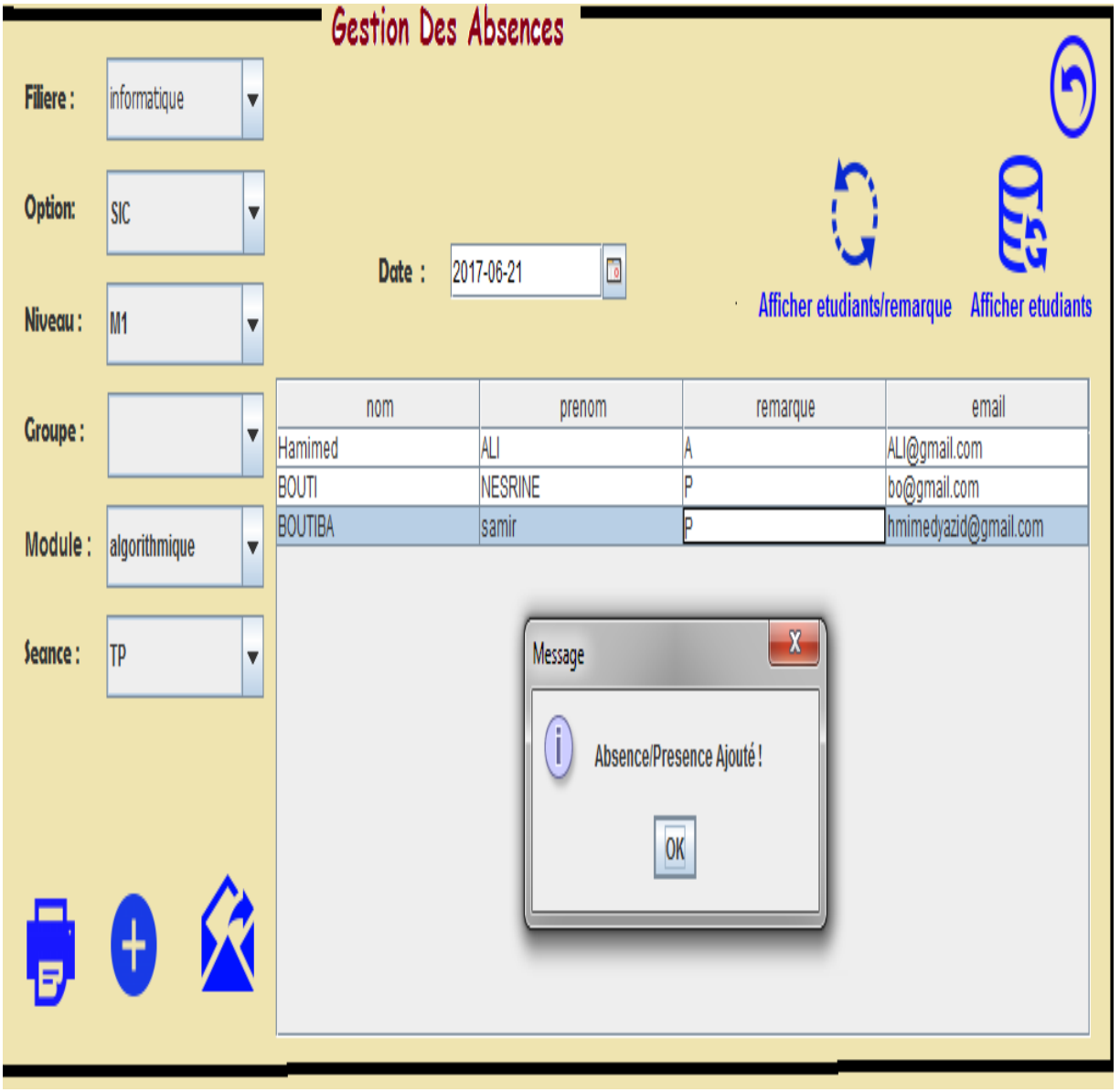


Figure 21 : Ajouter les absences

Modifier Absence

Cette interface permet a un agent de modifier l'absence d'un étudiant si seulement si l'étudiant justifier son absence, la justification sous forme image est enregistrer au niveau de serveur.

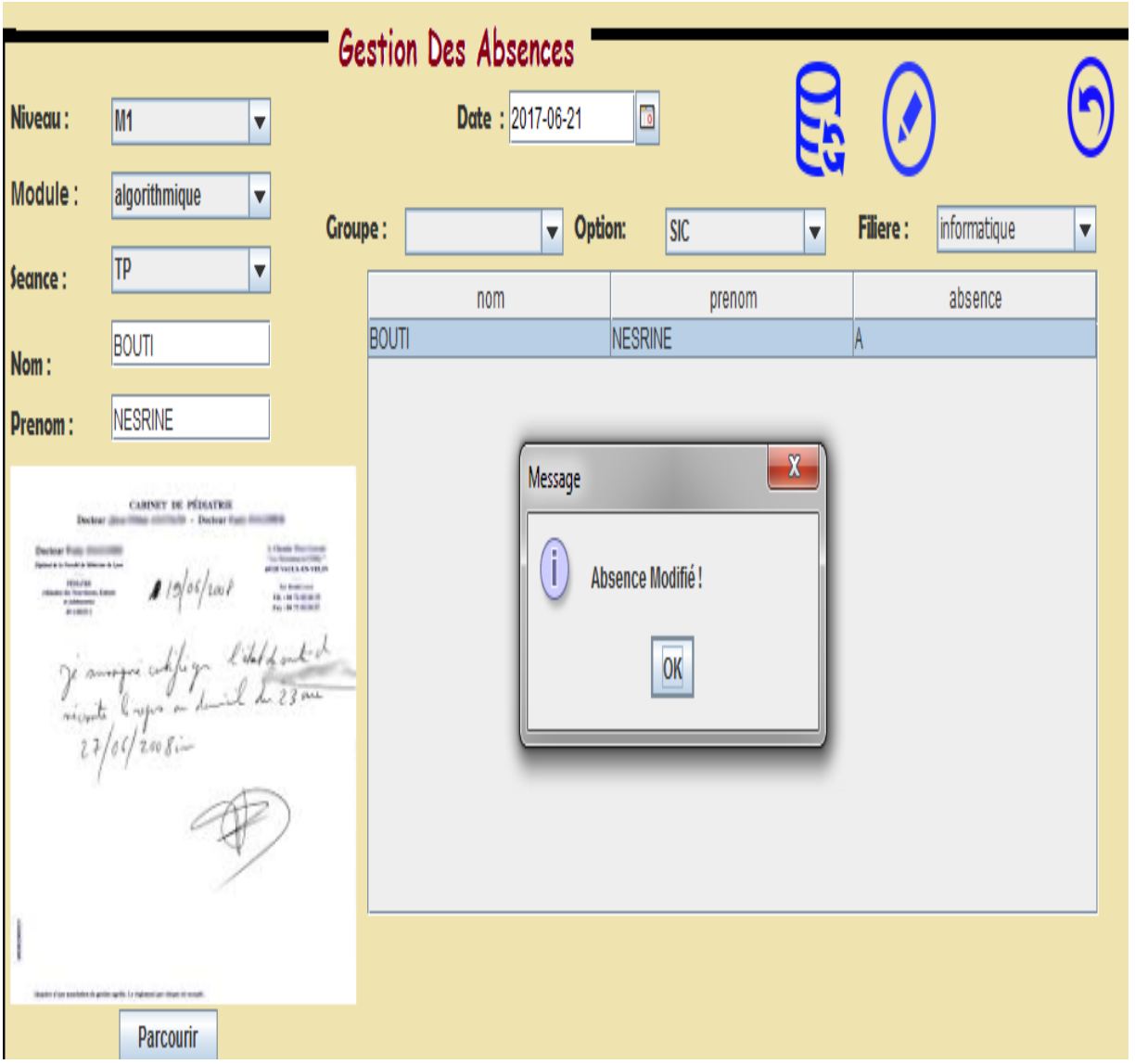


Figure 22 : Modifier absence

Recherche des absences

Cette IHM permet d'effectuer trois types de recherche :

- Etudiant/date/module.
- Date/module.
- Etudiant/module.

Le résultat de recherche est un tableau.

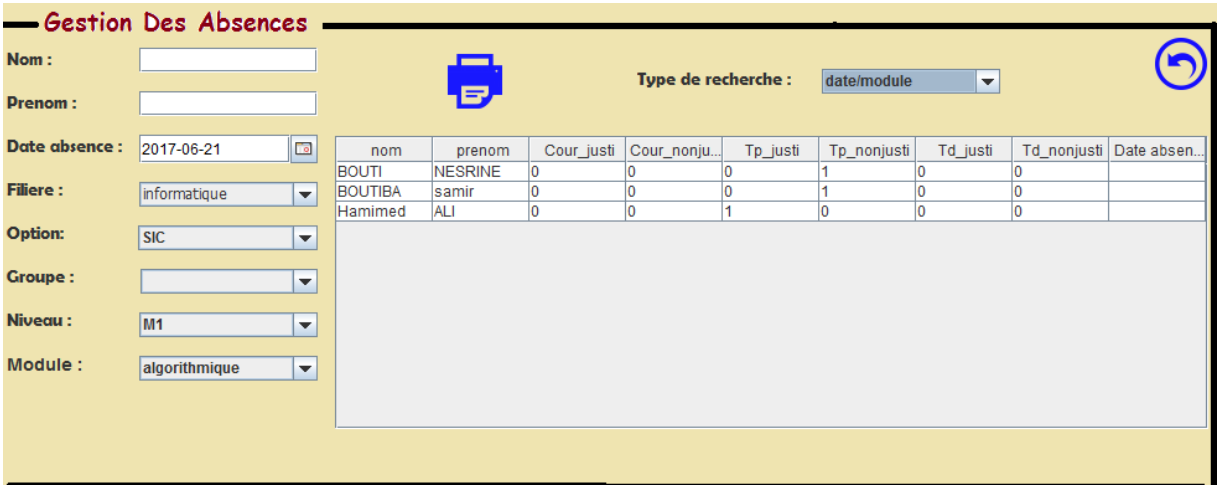


Figure 23 : Résultat de recherche par date/module

Exclu et Retard

Cette interface a pour but de trouver les étudiants exclus dans un module selon leur absence au niveau des séances de TD et TP.

Un étudiant soit exclu si seulement si leur nombre d'absence au TD et TP soit supérieur ou égal a 3 absence non justifié ou 5 justifié.

Notre application permet aussi de signaler les retards au niveau des séances**.**

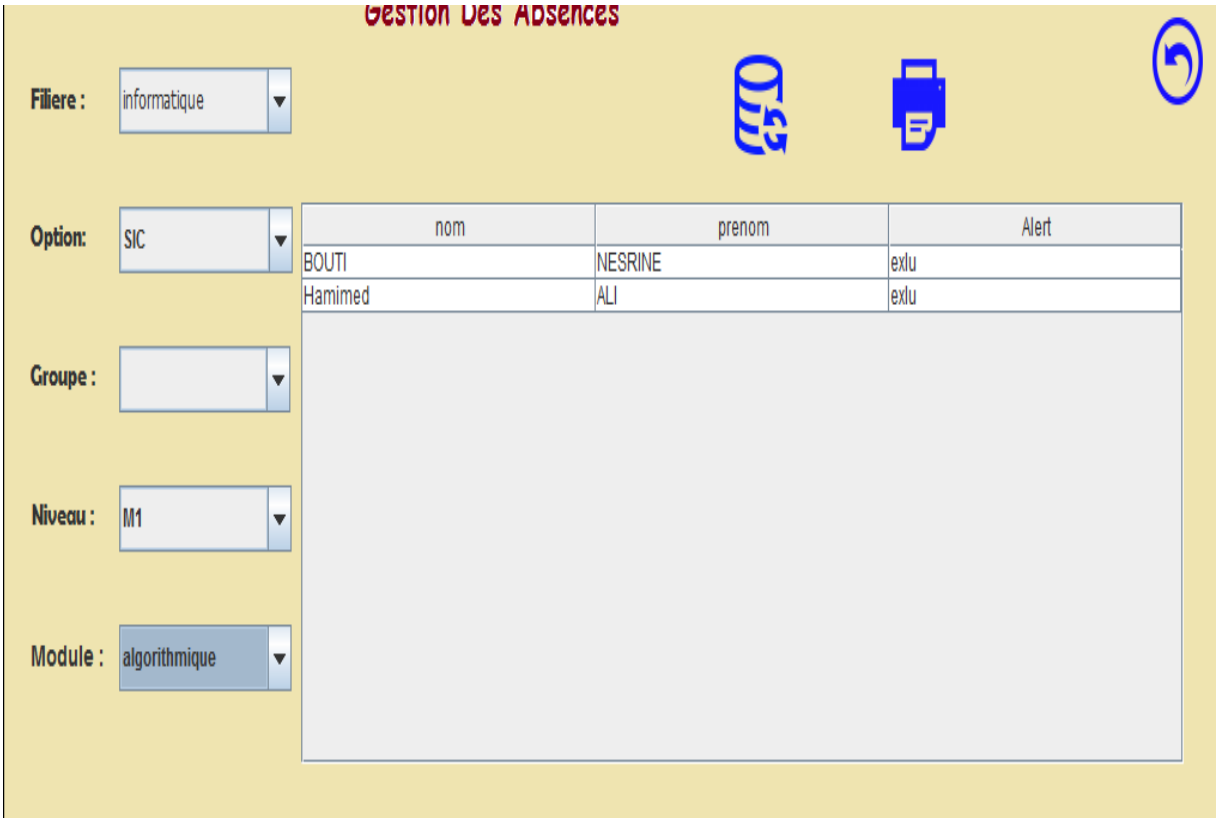


Figure 24 : Afficher les étudiants Exclus

Consultation des justificatifs

Cette interface permet au agent d'afficher les justificatifs d'un étudiant en forma d'image. L'ensemble de justificatif d'un étudiant sont affichée par date dans un tableau, un clic sur la ligne autorise l'affichage de l'image dans le label.

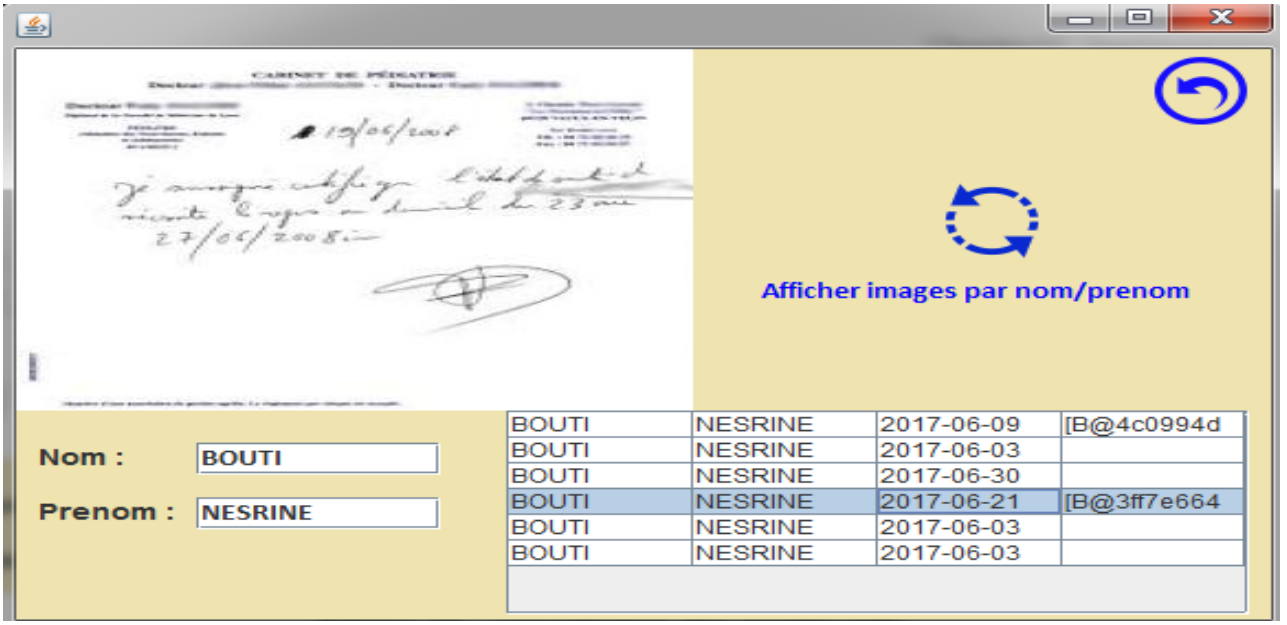


Figure 25 : consultation des justificatifs

III.9 La sécurité dans notre application

La **sécurité informatique**, est l'ensemble des moyens techniques, organisationnels, juridiques et humains nécessaires à la mise en place de moyen visant à empêcher l'utilisation non-autorisée, le mauvais usage, la modification ou le détournement du système d'information. Assurer la sécurité du système d'information est une activité du management du système d'information.

Aujourd'hui, la sécurité est un enjeu majeur pour les application ainsi que pour l'ensemble des acteurs qui l'entourent. Sa finalité sur le long terme est de maintenir la confiance des utilisateurs et des clients. La finalité sur le moyen terme est la cohérence de l'ensemble du système d'information. Sur le court terme, l'objectif est que chacun ait accès aux informations dont il a besoin.

Au niveau de notre application la sécurité se repose sur trois axes :

1-la sécurité de réseau

2-authentification

3- Mettre en place un **« security manager ».**

- **la sécurité de réseau**

Une politique de sécurité réseau est un document générique qui définit des règles à suivre pour les accès au réseau informatique et pour les flux autorisés ou non, détermine

comment les politiques sont appliquées et présente une partie de l'architecture de base de l'environnement de sécurité du réseau.

Nous avons utilisé un réseau ad hoc pour une communication sans fi**l** sans l'aide d'une infrastructure préexistante ou administration centralisée. L'accès a ce réseau que nous avons appelé « gestion » est sécurisée par un mot de passe.

- **Authentification**

L'authentification pour un système informatique est un processus permettant au système de s'assurer de la légitimité de la demande d'accès faite par une entité (être humain ou un autre système...) afin d'autoriser l'accès de cette entité à des ressources du système (systèmes, réseaux, applications…) conformément au paramétrage du contrôle d'accès . L'authentification permet donc, pour le système, de valider la légitimité de l'accès de l'entité, ensuite le système attribue à cette entité les données d'identité pour cette session (ces attributs sont détenus par le système ou peuvent être fournis par l'entité lors du processus d'authentification). C'est à partir des éléments issus de ces deux processus que l'accès aux ressources du système pourra être paramétré (contrôle d'accès).

Pour notre application le serveur et le client doit d'abord s'authentifier par un login et un mot de passe sauvegarder dans la base de donnée pour accéder au menu principal d'application client ou serveur.

- **RMISecurityManager**

Nous avons utilisé un fichier décrive la politique de permissions : « java.policy »,il définir la politique de sécurité afin de spécifier les action autorisée

III.10 Conclusion

Dans cette partie de notre projet, nous avons présentés les différents outils du développement de notre application ainsi que ses interfaces essentielles réalisé dans notre application web pour clarifier les étapes d'utilisation de notre application.

Conclusion Générale

Ce projet nous a donné la chance de découvrir les différentes architectures distribuées, en particulier l'architecture de type client/serveur et de se familiariser aussi avec l'outil java et en particulier avec le middleware java RMI.

Nous avons appris comment utiliser l'architecture client /serveur avec la méthode Java RMI pour développer notre application de Gestion Des Absences sur un réseau ad hoc.

L'utilisateur de notre application qui peut être un enseignant ou un agent dédié aura a sa porté une application facile a utiliser , d'une interface emviviale, il pourra a tous moment sauvegarde le suivi de ses étudiants, de plus avertir les étudiant et de manière automatique des que le nombre d'absence s'augmente.

Idéal que notre application soit intégré dans la gestion de scolarité.

Bibliographies

[1] Computer Systems Design and Architecture (International Edition) Andrew S. Tanenbaum 2004

[2] Architecture distribué Consulté le 05 2, 2017,sur université de Pau et des Pays de l'Adour UFR Sciences Pau : http://ecariou.perso.univ-pau.fr

[3] Architecture distribué Consulté le 05 4, 2017,sur Eric Leclercq : http://ireboot.ubourgogne.fr

[4]Au cœur de Java 2 volume 2 Fonctions avancées Cay S. Horstmann et Gary Cornell

[5] Architecture client serveur Consulté le 05 5, 2017 ,sur framasoft network : http://www.framasoft.net/article3991

[6] modes d'interaction client/serveur Consulté le 12 5, 2017 ,sur http://ibd.forge.imag.fr/IBD

[7]Esmond Pitt, Kathleen McNiff-java(TM).rmi_ The Remote Method Invocation Guide-Addison-Wesley Professional (2001)

[8] Java RMI William Grosso Publisher: O'ReillyFirst Edition October 2001

[9] Badr Benmammar, Introduction aux objets répartis Java RMI, Cours M2 RSD (Master Réseaux et Systèmes Distribués).2017

[10]RMI, J. (2011, 03 8). Java RMI . Consulté le 05 2, 2017, sur laboratoire d'informatique fondamentale de marseille : http://pageperso.lif.univ-mrs.fr

[11] Ghefir Mohamed El Amine, B. S. (2006). Développement d'une application distrubiée par la méthode Java RMI.

[12] CHRISTIANE Soutou, UML2 pour les bases de données , EYROLLES, 2015

[13] ArgoUml Consulté le 1 2, 2017, sur **http://www-igm.univ-mlv.fr**

[14] Pascal ROQUES, *UML 2 par la pratique étude de cas et exercices corrigés*., 2006

[15] *Développons en Java* Jean-Michel DOUDOUX.

[16] Eclipse Luna Consulté le 2 6, 2017, sur club des développeurs www.developpez.com

[17]Caractéristique de java Consulté le 12 6, 2017, sur club des développeurs www.developpez.com

[18] JDBC Consulté le 1 6, 2017, sur http://www.JDBC.pdf

[19] JDBC et SQL Consulté le 1 6, 2017, sur https://fr.wikipedia.org

[20] EasyPHP et MySQL Consulté le 12 6, 2017, sur https://www.jmdoudoux.fr

Annexe

Annexe 1

Exemple d'impression

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي

Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

Gestion des absences

Date :2017-06-30

Filiere :informatique

Option :SIC

Niveau :M1

Groupe :

Module :algorithmique

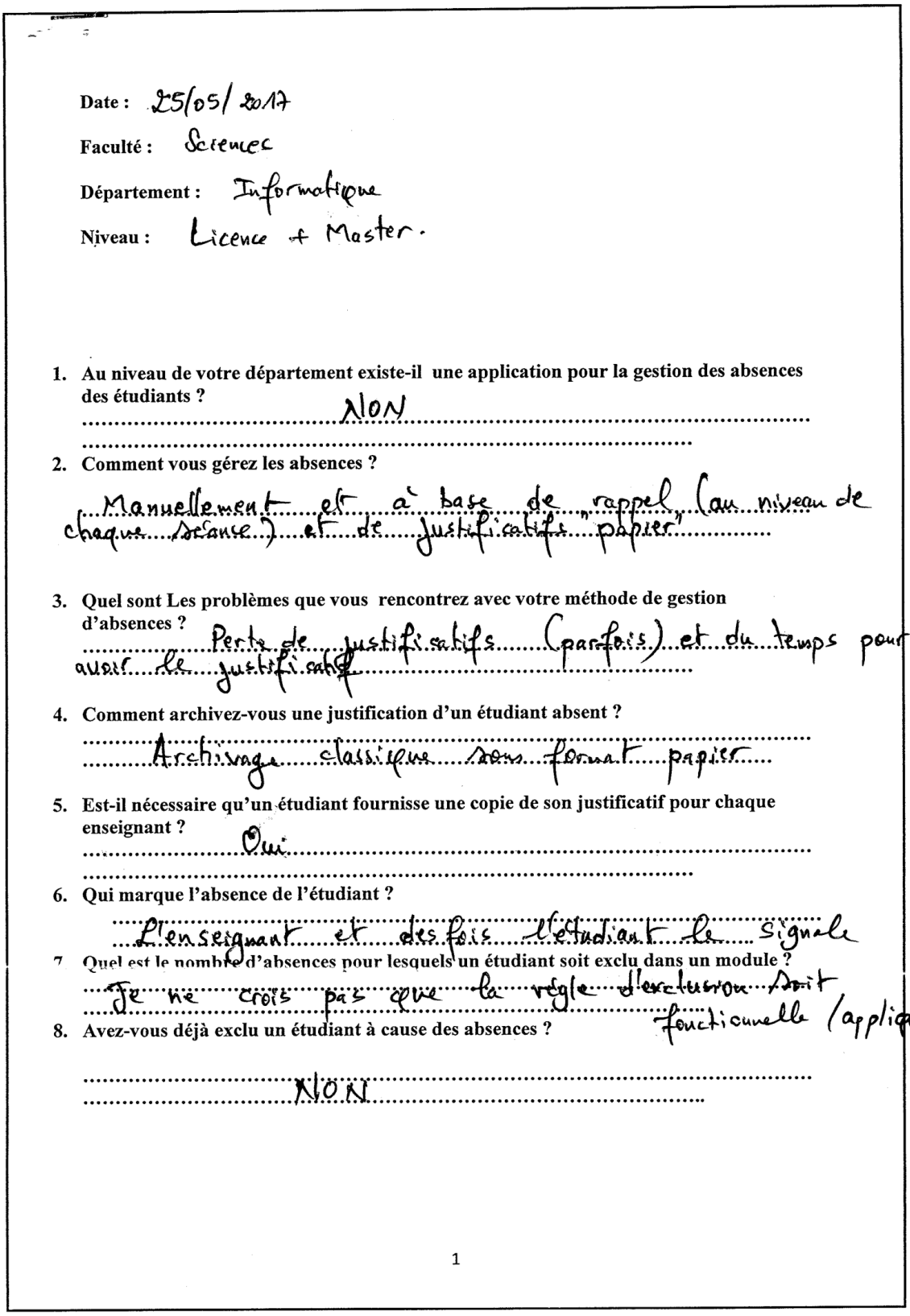
Seance :TD

Liste De Presences		
nom	prénom	Remarque
Hamimed	ALI	A
BOUTI	NESRINE	P
BOUTIBA	samir	P

Figure 26 : imprimer la liste de présence

Annexe 2

Exemple de questionnaire



Date : 25/05/2017

Faculté : Sciences

Département : Informatique

Niveau : Licence + Master.

1. Au niveau de votre département existe-il une application pour la gestion des absences des étudiants ?
   NON
2. Comment vous gérez les absences ?
   Manuellement et à base de rappel (au niveau de chaque séance) et de justificatifs "papier"
3. Quel sont Les problèmes que vous rencontrez avec votre méthode de gestion d'absences ?
   Perte de justificatifs (parfois) et du temps pour avoir le justificatif
4. Comment archivez-vous une justification d'un étudiant absent ?
   Archivage classique sous format papier
5. Est-il nécessaire qu'un étudiant fournisse une copie de son justificatif pour chaque enseignant ?
   Oui
6. Qui marque l'absence de l'étudiant ?
   L'enseignant et des fois l'étudiant le signale
7. Quel est le nombre d'absences pour lesquels un étudiant soit exclu dans un module ?
   Je ne crois pas que la règle d'exclusion doit fonctionnelle (applique
8. Avez-vous déjà exclu un étudiant à cause des absences ?
   NON

1

Figure 27 : la partie1 du questionnaire

9. **Est-ce que vous appliquez cette procédure si un étudiant dépasse le nombre d'absence ?**

NON

10. **Est-ce que les étudiants qui ont plus d'absence au niveau des cours sont pénalisés ?**

NON, plutôt au niveau des TD et TP.

11. **Est que les retards sont pris en considération ?**

NON

12. **Comment jugez-vous les retards ?**

?

13. **Pensez-vous qu'il serait utile d'informatiser la gestion des absences ?**

Oui, on aimerait bien avoir une application pour
(1) Signaler les absences, (2) Uploader les justificatifs,
(3) Calculer la note de présence automatiquement.

2

Figure 28 : la partie2 du questionnaire

Annexe 3

Création du réseau ad hoc

A partir du **menu Démarrer**, recherchez le **Centre Réseau et partage** puis ouvrez ce module.

Dans la fenêtre qui apparaît, cliquez sur **Configurer une nouvelle connexion ou un nouveau réseau**.

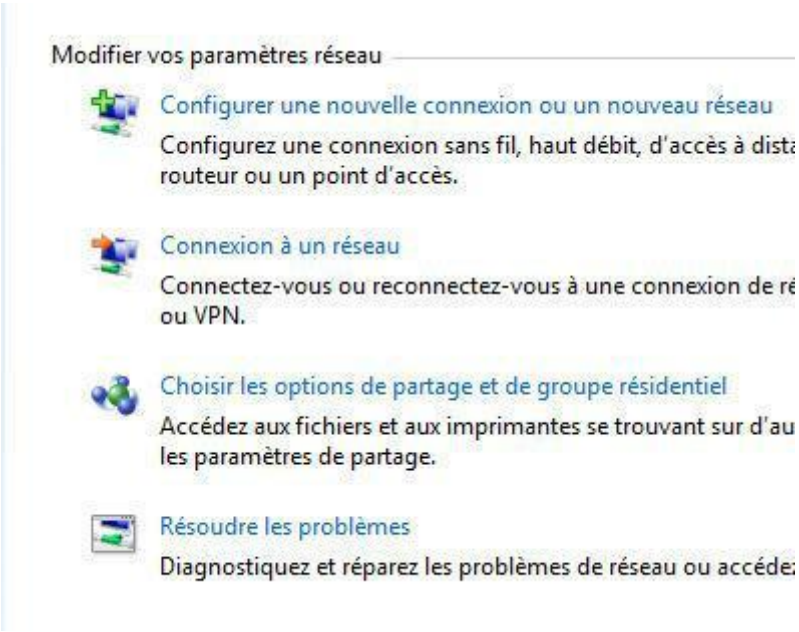


Figure 29 : Aperçu de la création d'un réseau ad hoc via le Centre Réseau et partage

Un assistant de création de connexion va maintenant vous guider pas à pas.

Sélectionnez le dernier choix dans la liste : **Configurer un réseau sans fil ad hoc (ordinateur à ordinateur)**.

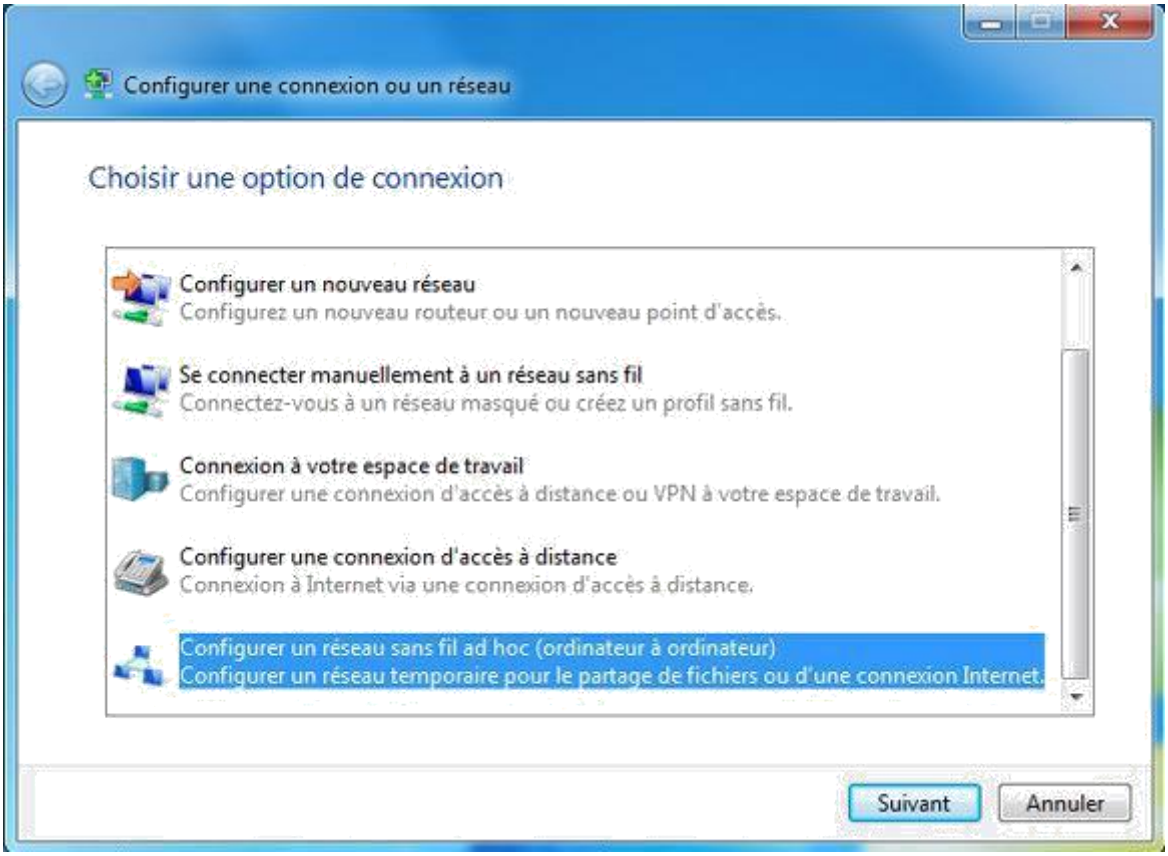


Figure 30 **:** Aperçu d'assistant de création d'un réseau ad hoc

Un écran d'information vous indique alors en quoi consiste un réseau ad hoc. Retenez que les 2 PC ne doivent pas être éloignés de plus de 10 mètres et qu'il s'agit d'un mode temporaire.

Figure 31 : Aperçu d'informations relatives à la création d'un réseau poste à poste

Nous allons désormais attribuer un nom au réseau et choisir les options de sécurité :

- **Nom réseau** : indiquez le nom que portera le réseau. Il sera ensuite détecté comme tel via WIFI ;
- **Type de sécurité** : optez pour **WPA2 – Personnel** qui offre le meilleur niveau de protection. N'oubliez pas que tout le monde verra ce nouveau réseau et qu'il sera donc accessible dans un environnement proche : il nous faut donc le sécuriser ;
- **Clé de sécurité** : indiquez un mot de passe qui sera ensuite crypté selon la norme précédemment choisie. Décochez **Masquer les caractères** pour visualiser la clé lors de la saisie ;

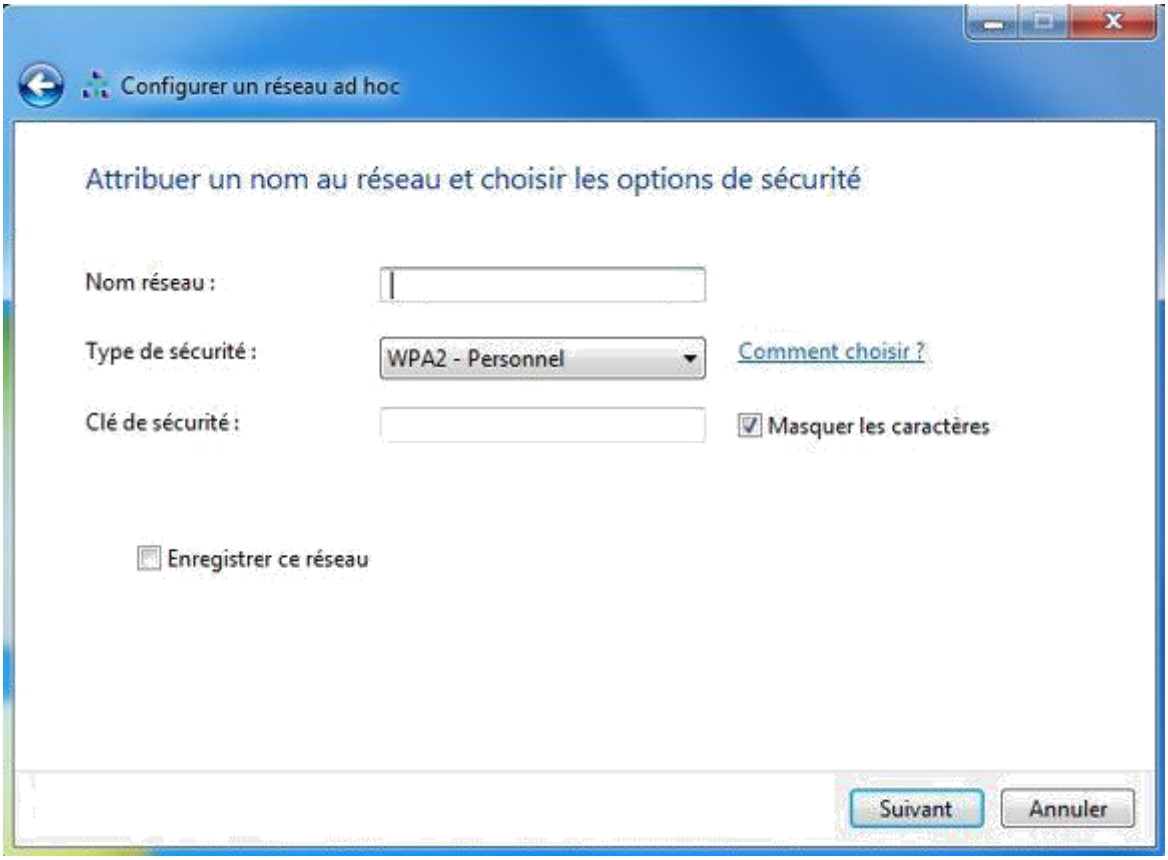


Figure 32 : **Aperçu de configuration de la sécurité du réseau ad hoc**

Cochez **Enregistrer ce réseau** pour une réutilisation ultérieure puis cliquez sur **Suivant**.

Patientez quelques secondes. Votre réseau ad hoc est à présent

opérationnel. Il vous permettra de :

- Partager des fichiers ;
- Partager votre connexion Internet en cliquant sur le bouton **Activer le partage de connexion Internet** ;

Il ne reste plus qu'à fournir la clé de sécurité à votre voisin pour qu'il puisse se connecter à votre machine et bénéficier des ressources que vous partagez.

Résumé

Ce travail consiste à développer une application distribuée de gestion des absences des étudiants au niveau de la faculté des sciences de notre université. Le but étant d'informatiser cette tache fastidieuse et de grande importance. Cela permettra un bon suivi de l'assiduité de chaque étudiant, et facilitera la sauvegarde des justificatifs d'absences fournis par les étudiants. L'utilisateur de notre application sera averti dès que leur absence non justifier dans un module sera supérieur ou égale a 2 un email est envoyé aux étudiants dont le nombre d'absences égal e a 2

Ce travail est partitionné en trois parties, premièrement nous avons détaillé le architecture client serveur et le java RMI. Deuxièmement, l'analyse et la conception de notre système et finalement la réalisation de notre application de gestion des absences.

MOTS CLES : absences, gestion, client serveur , Java, RMI

ملخص

في إطار أطروحة الماستر قمنا بانشاء تطبيق إدارة غياب باستعمال **java RMI**. هذا التطبيق العميل الخادم يسمح وضع وبالتالي الحصول على وقت اقل للبحث. منظومة كمبيوتر لإدارة غياب الطلاب في كلية العلوم حيت يهدف لتخزين مبررات الغياب التي يقدمها الطلاب كما اته يتم من خلال هدا التطبيق اندار الطلاب الدين قد بلغو غيابيين غير مبررين.

وينقسم هذا العمل إلى ثلاثة أجزاء، أولا بنية الخادم والعميل **java RMI**. ثانيا، تحليل وتصميم نظامنا، وفي نهاية المطاف تحقيق تطبيق إدارة غيابنا.

الكلمات الرئيسية: الغياب والإدارة وخدمة العملاءjava RMI

Abstract

This work involves the development of a distributed student absentee management application at the level of the Faculty of Science at our university. The aim is to computerize this tedious task of great importance. This will allow a good follow-up of the attendance of each student, and will facilitate the safeguarding of the justifications of absences provided by the students. The user of our application will be notified as soon as their absence not justifying in a module will be greater or equal to 2 an email is sent to the students whose absences equal to 2.

This work is partitioned into three parts; first we have detailed the client server architecture and the java RMI. Second, the analysis and design of our system and finally the realization of our application of absentee management.

KEY WORDS: absences, management, client server, Java, RMI