**Université Sidi Mohamed Ben Abdellah**

**Ecole Supérieure de Technologie -Fès**

***Mémoire de fin d'études***

***Présenté pour l’obtention du***

***Diplôme Universitaire de Technologie en***

***Réseaux et Télécommunication***

***Développement d’une application temps réel de messagerie Client\_serveur***

**Réalisé par : Encadré Par :**

**EL AMARTI MOAD M. ED-DAOUDY**

**DYBELE GBENDA DYLAN JUNIOR M. MAALMI**

***Année Universi******taire : 2019 – 2020***

*Remerciements*

*Concernant notre projet de fin d’étude, nous adressons nos sincères remerciements en premier lieu au bon Dieu, pour le support qu’il nous a apporté dans la conception de notre projet.*

*Nos remerciements s’adressent également à l’administration et aux professeurs de L’EST, Pour les moyens qu’ils ont mis à notre disposition afin d’élaborer ce travail.*

*Et je dédie ce modeste travail à L’équipe Administratif de l’EST de Fès, à mes ami(e)s, à mes frères et sœurs et tout spécialement à mes chers parents.*

*Nous remercions toutes les personnes qui, d’une quelconque manière, nous ont apporté leur amitié, leur attention, leurs encouragements, leur appui et leur assistance pour que nous puissions mener à terme ce travail*

*Pour finir, nous remercions les membres du jury qui ont accepté d’évaluer notre projet. Nous leurs présentons toute nos gratitudes et nos profonds respects.*

*MOAD & DYLAN*

### Liste des figures

### Figure 1 : Client / Serveur

### Figure 2 : Modèle Client / Serveur

### Figure 3 : Architecture 1 - tiers

### Figure 4 : Architecture 2 - tiers

### Figure 5 : Architecture 3 - tiers

### Figure 6 : Architecture n - tiers

### Figure 7 : Comparaison mode connecté et non connecté

### Figure 8 : Echange en protocole TCP

### Figure 9 : Comparaison entre protocole TCP et UDP

### Figure 10 : Communication en mode connecté avec les sockets

### Figure 11 : Communication en mode non connecté avec les sockets

### ***Sommaire***

### Remerciements

### Liste des figures

### Introduction générale

# **Chapitre I**

## I. Généralité sur le terme client/ serveur

## II. Les différentes architectures client / serveur

## III. Les avantages et inconvénients de l’environnement client/ serveur

### 1. Avantages d’un environnement client / serveur

### 2. Avantages d’un environnement client / serveur

### IV. Les modes de connexion, les protocoles et les sockets

### 1. Les modes de connexion

### 2. Les protocoles TCP et UDP

### V. Les différents concepts

### 1.Les sockets

# **Chapitre II**

## I. Introduction

## II. L’environnement de travail

## 1. JAVA

## 2.Eclipse

## 3. AWT et SWING

## 4. JDBC (Java DataBase Connectivity)

## 5.UML

6.XAMPP Control Panel

7.My SQL

## III. Présentation de l’application

## IV. Modélisation UML

### 1.Diagrame de classe

### 2.Diagrame de séquence

### 3.Diagrame de cas d’utilisation

## V. Les interfaces de l’application

1. Coté Serveur
2. Coté Client

## VI. Fonctionnement de l’application

## L’utilisation de l’application

## VII. Conclusion

### Introduction générale

La communication occupe une place très importante dans la société d’aujourd’hui, à ce stade, il ne fait désormais plus aucun doute que les technologies de l'information et de la communication représentent une révolution importante ainsi qu’innovante qui a marqué la vie de l'humanité en ce siècle.

En effet, ces technologies viennent nous apporter de multiples conforts, car ils ont révolutionné le travail des individus par leur capacité de traitement d'information, d'une part, et de rapprochement des dimensions espace/temps, d'une autre.

Pour rendre hommage à cela, nous avons décidé de porter notre projet sur le développement d’une application de messagerie Client/ Serveur qui a pour but de centraliser l’information, ainsi que de la sécuriser le mieux possible. Pour la petite histoire c’est en 1994 que l’architecture client-serveur s’implante sur le marché, non seulement cette architecture rend possible l’amélioration de la capacité de stockage des données, mais elle contribue à augmenter considérablement la rapidité de traitement des postes de travail. En effet, par la structure du serveur qui contient plusieurs unités de traitement (CPU) permettant de gérer les postes clients, la capacité ainsi que la mémoire sont accrues. La progression de l’architecture client-serveur ne sera pas linéaire. En effet, cette technologie se développera à un rythme des plus accélérés. On peut affirmer qu’elle a connu une croissance exponentielle dans tous les domaines d’activités tels que la gestion de base de données, les systèmes de messagerie, web, Internet, les systèmes de partages des données etc ... depuis sa création jusqu’aujourd’hui.

Ainsi notre projet s'articulera donc autour de deux chapitres :

Chapitre 1 : Etude de l’architecture Client/ Serveur et ses fonctionnalités.

Dans ce chapitre, nous présenterons le sens du terme Clients / Serveur, ses avantages ainsi que ses inconvénients et également les différentes architectures Client / Serveur existant.

Chapitre 2 : Application Et Outils utilisés

Ce chapitre, s'intéresse à la présentation et réalisation de l’application, il contient les outilles utilisés, l’environnement du travail, le fonctionnement de l’application et les principales interfaces.

### Chapitre I

### I.Généralité sur le terme Client/ Serveur

L'environnement client/serveur désigne un mode de communication organisé par l'intermédiaire d'un réseau et d'un interface Web entre plusieurs ordinateurs cela signifie que des machines clientes (machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en termes de capacités d'entrées-sorties, qui leur fournit des services, ces services sont généralement des programmes fournissant des données telles que l'heure, des fichiers, une connexion, etc.

Le système fonctionne de telle sorte que :

* Le client émet une requête vers le serveur grâce à son [adresse IP](https://www.commentcamarche.net/contents/523-adresse-ip) et le [port](https://www.commentcamarche.net/contents/528-port-ports-tcp-ip), qui désigne un service particulier du serveur
* Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine cliente et son port

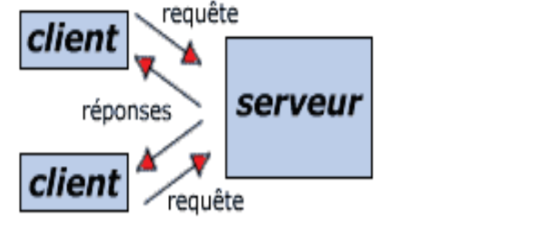


Figure 1 : Client / Serveur

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

On retrouve aussi le terme serveur qui est un ordinateur spécialisé dans la fourniture et le stockage des ressources partagées des utilisateurs réseau et est détentrice des caractéristiques suivantes :

* Fournisseur de service ;
* Il est à l'écoute, prêt à répondre aux requêtes envoyées par des clients ;
* Dès qu'une requête lui parvient, il la traite et envoie une réponse ;
* Traitement de plusieurs clients simultanément.

Le client quant à lui et un ordinateur qui accède aux ressources partagées fournies par un serveur du réseau.

Caractéristiques d'un client :

* Consommateur de service ;
* Il établit une connexion au serveur ; · Il envoie des requêtes au serveur ;
* Il attend et reçoit les réponses du serveur

Dans les termes à retenir il y’a également la requête qui est un message transmis par un client à un serveur décrivant l’opération à exécuter pour le compte du client.

Pour finir, le terme réponse ici est un 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.

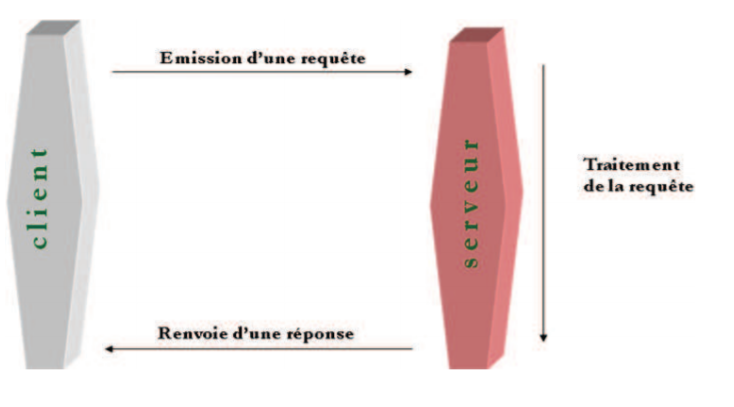


Figure 2 : Modèle Client / Serveur

### II.Les différentes architectures Client / Serveur

Une application informatique comme c’est le cas de notre application de messagerie de Client / Serveur, peut être découpée en trois niveaux :

* 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 effectués 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.

Plusieurs architectures ou environnements existent, ces architectures diffèrent selon les critères et par les caractéristiques du réseau ou du client qu'ils contiennent.

On retrouve donc :

***Architecture 1-tiers***

Dans une application un tiers les couches applicatives sont liées et s’exécutent sur le même ordinateur, elle est surtout caractérisée par le fait que chaque ordinateur connecté au réseau est susceptible de jouer tour à tour le rôle de client et celui de serveur.

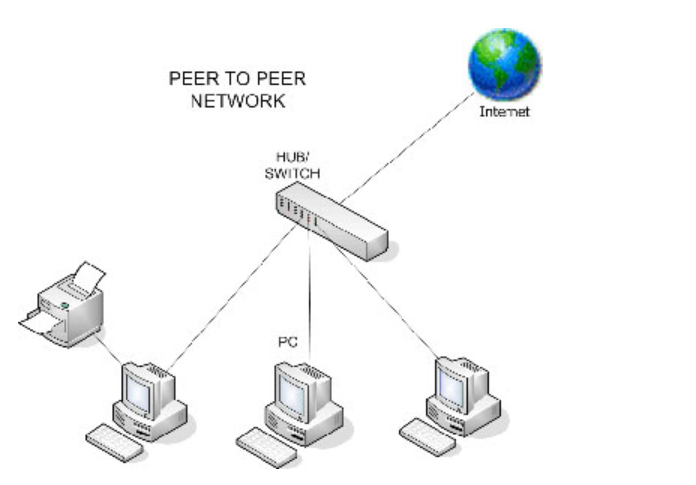


Figure 3 : Architecture 1 – tiers

***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 traverse un réseau reliant les deux machines (client et serveur), il met en œuvre des mécanismes relativement complexe qui sont, en général, pris en charge par un middleware.

Le cas typique de cette architecture est une application de gestion fonctionnement sous Windows ou Linux et exploitant un SGBD centralisé.

S’exécutant le plus souvent sur un serveur dédié, ce dernier est interrogé en utilisant un langage de requête comme SQL. Ce type d’application permet de tirer une partie de la puissance des ordinateurs déployés en réseau pour fournir à l’utilisateur une interface riche, tout en garantissant la cohérence des données, qui restent gérées de façon centralisée. 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.

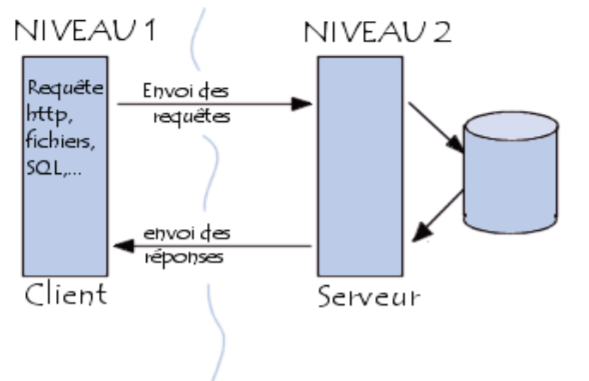


Figure 4 : Architecture 2 – tiers

***Architecture 3-tiers***

Pour résoudre les limitations du client/serveur 2-tiers tout en conservant ses avantages, on a cherché une architecture plus évoluée, facilitant les forts déploiements à moindre coût, La réponse est apportée par les architectures distribuées, en utilisant un poste client simple communicant avec le serveur par le biais d’un protocole standard. 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…) 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.

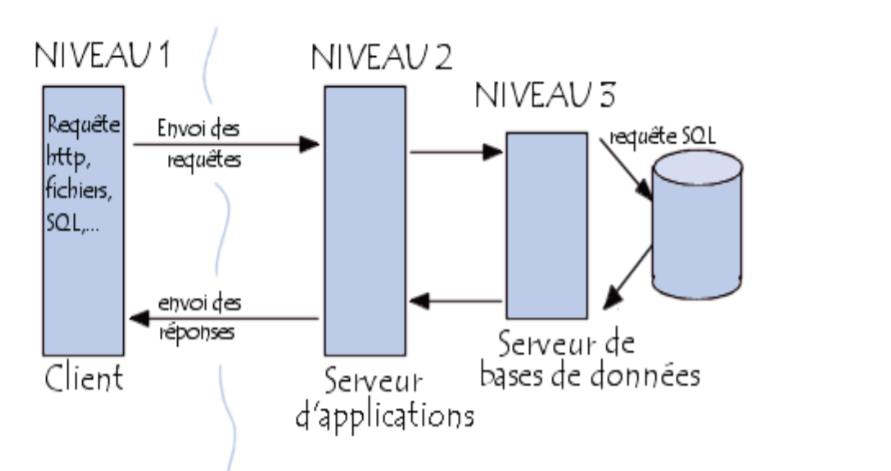


Figure 5 : Architecture 3 – tiers

***Architecture 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 standards entre application et le concept Middleware objet

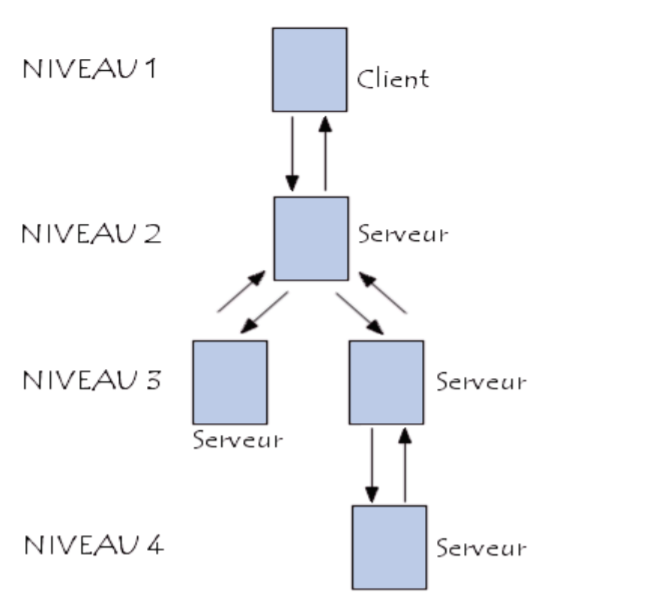


Figure 6 : Architecture n – tiers

### III.Les avantages et inconvénients de l’environnement Client/ Serveur

### 1. Avantages d’un environnement Client / Serveur

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

* + Des ressources centralisées :

Etant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction.

* + Une meilleure sécurité :

Car le nombre de points d'entrée permettant l'accès aux données est moins important.

* + Une administration au niveau serveur :

Les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés.

* + Un réseau évolutif :

Grâce à cette architecture il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modification majeure.

### 2. Avantages d’un environnement Client / Serveur

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

* + Un coût élevé dû à la technicité du serveur.
  + Un maillon faible :

Le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui.

### IV.Les modes de connexion et les protocoles

### 1.Les modes de connexion

La communication dans une architecture Client / Serveur peut être établie de deux manières entre deux appareils ou plus, qui sont orientés connexion (mode connecté) et sans connexion (mode non connecté). Les couches réseau peuvent offrir ces deux types de services à leur couche prédécesseur pour le transfert de données. Les services orientés connexion impliquent l’établissement et la fin de connexion alors que les services sans connexion ne nécessitent aucun processus de création et de terminaison de connexion pour le transfert de données.  
   
Une autre différence entre les services orientés connexion et sans connexion est que la communication orientée connexion utilise un flux de données et il est vulnérable à la défaillance du routeur tandis que la communication sans connexion utilise des messages et il est robuste aux pannes de routeur.

***Mode connecté***

Le service orienté connexion ou le mode connecté est conforme au système téléphonique qui nécessite des entités de communication pour établir une connexion avant d’envoyer des données. Le protocole [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) fournit des services orientés connexion, tout comme le matériel ATM, Frame Relay et MPLS. Il utilise un processus d’établissement de liaison pour établir la connexion entre l’expéditeur et le destinataire.

Plus précisément, il met en place une connexion, utilise cette dernière puis met fin à la connexion.

La fiabilité est obtenue en demandant au destinataire de reconnaître chaque message (ACK). Il y a le séquençage et contrôle de flux, c’est la raison pour laquelle les paquets reçus à la réception sont toujours en ordre. Il utilise la commutation de circuit pour la transmission de données.

En mode connecté, la communication entre un client et un serveur est précédée d’une connexion et suivie d’une fermeture :

* Facilite la gestion d’état
* Meilleur contrôle des arrivées/départs de clients
* Uniquement communication unicast
* Plus lent au démarrage

***Mode non connecté***

Le service sans connexion ou le mode non connecté est analogue au système postal. Dans lequel chaque paquet de données (généralement connu sous le nom de datagramme) est transmis de la source à la destination directement. Chaque paquet est traité comme une entité individuelle, ce qui permet aux entités de communication d’envoyer des données avant d’établir une communication. Chaque paquet porte une adresse de destination pour identifier le destinataire prévu.

Les paquets ne suivent pas un chemin fixe qui est la raison pour laquelle les paquets reçus à la fin du récepteur peuvent être hors service. Il utilise la commutation de paquets pour la transmission de données.

La plupart du matériel réseau, le protocole Internet (IP) et le protocole [UDP (User Datagram Protocol)](https://waytolearnx.com/2019/06/protocole-udp.html) fournissent un service sans connexion.

En mode non connecté, les messages sont envoyés librement :

* Plus facile à mettre en œuvre
* Plus rapide au démarrage

***Différence entre le mode connecté et le mode non connecté***

* Il y a une exigence de connexion préalable pour la communication dans les services orientés connexion (mode connecté), en revanche, elle n’est pas nécessaire dans les services sans connexion (mode non connecté).
* La fiabilité est plus axée sur les services orientés connexion que sur les services sans connexion.
* L’encombrement du trafic est plus important dans les services sans connexion, alors que son occurrence est rare dans les services orientés connexion.
* Dans les services orientés connexion, l’ordre des paquets reçus à la destination est identique à celui envoyé depuis la source. Tandis que, l’ordre pourrait changer dans les services sans connexion.
* Tous les paquets suivent le même chemin dans les services orientés connexion alors que les paquets suivent un chemin aléatoire pour atteindre la destination dans les services sans connexion.
* Le service orienté connexion est approprié pour une communication longue et stable, tandis que le service sans connexion est adapté pour une transmission par salves.
* Dans les services orientés connexion, l’expéditeur et le destinataire sont synchronisés entre eux, ce qui n’est pas le cas des services sans connexion.
* D’autre part, les services orientés connexion utilisent la commutation de circuits, tandis que la commutation de paquets est utilisée dans les services sans connexion.
* Le besoin de la bande passante est plus élevé dans les services orientés connexion, alors qu’il est faible dans les services sans connexion.

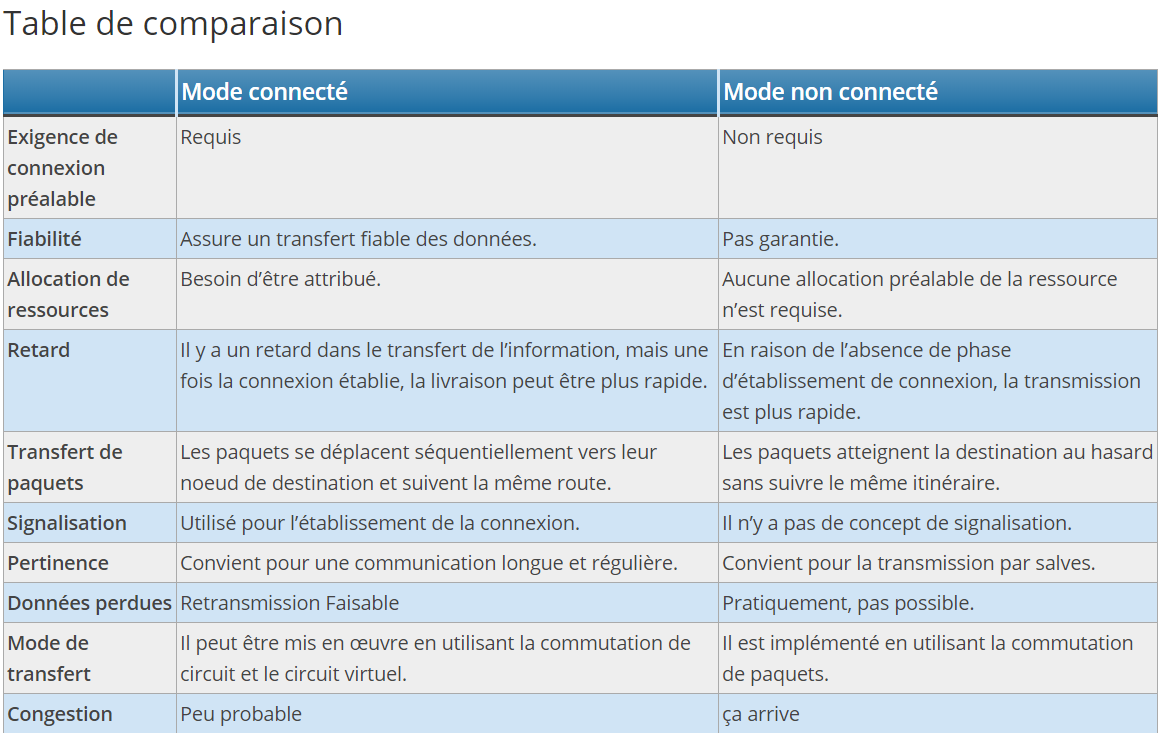


Figure 7: Comparaison mode connecté non connecté

### 2.Les protocoles TCP et UDP

TCP et [UDP](https://waytolearnx.com/2019/06/protocole-udp.html) se trouvent dans la quatrième couche du modèle OSI qui est la couche de transport juste au-dessus de la couche IP. [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) et [UDP](https://waytolearnx.com/2019/06/protocole-udp.html) les deux supportent la transmission de données de deux manières différentes, [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) est en mode orienté connexion et [UDP](https://waytolearnx.com/2019/06/protocole-udp.html) est en mode non-connecté.

Lors de la transmission d’une paquet, il y a deux contraintes majeures: la fiabilité et la latence. La fiabilité garantie la livraison du paquet et la latence garantie le temps opportun pour la livraison du paquet. Les deux ne peuvent pas être atteints en même temps mais peuvent être optimisés.

Afin d’initier une communication de données entre deux nœuds, l’expéditeur doit connaître l’adresses IP du récepteur ainsi que le numéro de port. L’adresse IP consiste à acheminer le paquet et le numéro de port est de transférer le paquet à la bonne destination.

les applications [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) écoutent sur un port pour accepter les connexions [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html). De même que les applications [UDP](https://waytolearnx.com/2019/06/protocole-udp.html) écoutent également sur un port pour fournir des services [UDP](https://waytolearnx.com/2019/06/protocole-udp.html).

***Le protocole TCP***

[TCP](https://waytolearnx.com/2019/06/protocole-tcp.html)(Transport Layer Protocol) est un protocole fiable de bout en bout orienté connexion pour garantir la transmission de données. Certaines des principales caractéristiques de [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) sont la détection d’erreur, le démarrage lent, le contrôle de flux et le contrôle de congestion.

[TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) est un mécanisme de transport fiable donc il sera utilisé là où la livraison de paquets est obligatoire même dans les congestions. Exemple typique pour les applications [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) et les numéros de port :

Port de données FTP (20), port de contrôle FTP (21), SSH (222), Telnet (23), Mail (25), DNS (53), HTTP (80), POP3 (110), SNMP (161) et HTTPS (443). Ce sont des applications [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) bien connues.

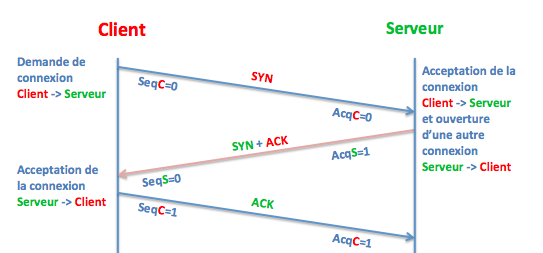
******

Figure 8 : Echange en protocole TCP

***Le protocole UDP***

[UDP](https://waytolearnx.com/2019/06/protocole-udp.html)(User Datagram Protocol) est un protocole de transmission simple qui fournit un service non fiable. Cela ne signifie pas que [UDP](https://waytolearnx.com/2019/06/protocole-udp.html) ne fournira pas les données mais il n’y a pas de mécanismes pour surveiller le contrôle de congestion ou la perte de paquets, etc. Comme c’est simple, cela évite le traitement de la congestion. Les applications en temps réel utilisent principalement [UDP](https://waytolearnx.com/2019/06/protocole-udp.html) car la suppression des paquets est préférable. Un exemple typique est celui des flux de média (Streaming).

###### ***Différence entre le TCP et UDP***

* [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) est en mode orienté connexion et fiable, tandis que [UDP](https://waytolearnx.com/2019/06/protocole-udp.html) est en mode non-connecté et peu fiable.
* [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) nécessite plus de traitement au niveau de l’interface réseau, ce qui n’est pas le cas en [UDP](https://waytolearnx.com/2019/06/protocole-udp.html).
* [TCP](https://waytolearnx.com/2019/06/protocole-tcp.html) utilise, handshake 3 voies, le contrôle de la congestion, le contrôle de flux et d’autres mécanismes pour assurer la transmission fiable.
* [UDP](https://waytolearnx.com/2019/06/protocole-udp.html) est principalement utilisé dans les cas où le retard de paquet est plus sérieux que la perte de paquet. (Applications en temps réel)

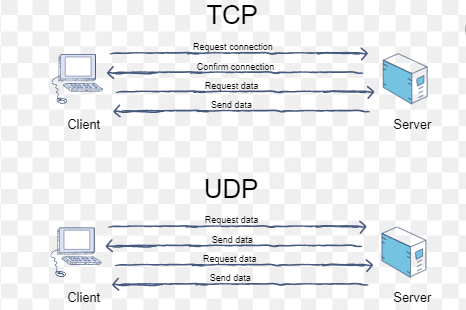
******

Figure 9 : Comparaison protocole TCP et UDP

### V. Les différents concepts

### 1.Les sockets

***Définition***

Un socket est connu comme un type de logiciel qui agit comme un point d'extrémité qui fonctionne en établissant une liaison de communication réseau bidirectionnelle entre l'extrémité du serveur et le programme de réception du [client](https://www.speedcheck.org/fr/wiki/client/).

Ces sockets sont réalisés et mobilisés en même temps qu'un ensemble de requêtes de programmation identifiées comme appels de fonction, qui est techniquement appelé interface de programmation d'application (API). Un socket est capable de simplifier le fonctionnement d'un programme car les programmeurs n'ont plus qu'à se soucier de manipuler les fonctions du socket, ce qui leur permet de compter sur le système d'exploitation pour transporter correctement les messages sur le réseau

*Fonctionnalité*

Typiquement, un socket respecte un flux spécifique d'événements pour qu'elle fonctionne. Pour un modèle client-serveur orienté connexion, le socket du processus serveur attend la demande d'un client. Pour ce faire, le serveur doit d'abord établir une adresse que les clients peuvent utiliser pour trouver et se connecter au serveur. Lorsqu'une connexion est établie avec succès, le serveur attend que les clients demandent un service. L'échange de données client-serveur aura lieu si le client se connecte au serveur via le socket. Le serveur répondra alors à la demande du client et lui enverra une réponse.

La plupart du temps, les U[RL](https://www.speedcheck.org/fr/wiki/rl/) et leurs connexions sont utilisées pour accéder à Internet. Les programmes nécessiteront un simple lien de communication entre le client et le serveur du programme. Ce rôle est associé à un socket qui aidera à lier les extrémités client et serveur du programme. Si un client commence à établir une communication avec le serveur, une connexion fiable entre le serveur et le client sera établie par le canal de communication [TCP](https://www.speedcheck.org/fr/wiki/tcp/). Avec ce type de communication, le client et le serveur peuvent lire ou écrire sur des sockets qui sont liés à un canal de communication spécifique.

*Déroulement de la communication avec les sockets*

Comme dans le cas de l'ouverture d'un fichier, la communication par socket utilise un descripteur pour désigner la connexion sur laquelle on envoie ou reçoit les données. Ainsi la première opération à effectuer consiste à appeler une fonction créant un socket et retournant un descripteur (un entier) identifiant de manière unique la connexion. Ainsi ce descripteur est passé en paramètres des fonctions permettant d'envoyer ou recevoir des informations à travers le socket.

L'ouverture d'un socket se fait en deux étapes :

* La création d'un socket et de son descripteur par la fonction *socket()*
* La fonction *bind()* permet de spécifier le type de communication associé au socket (protocole TCP ou UDP)
* En mode connecté

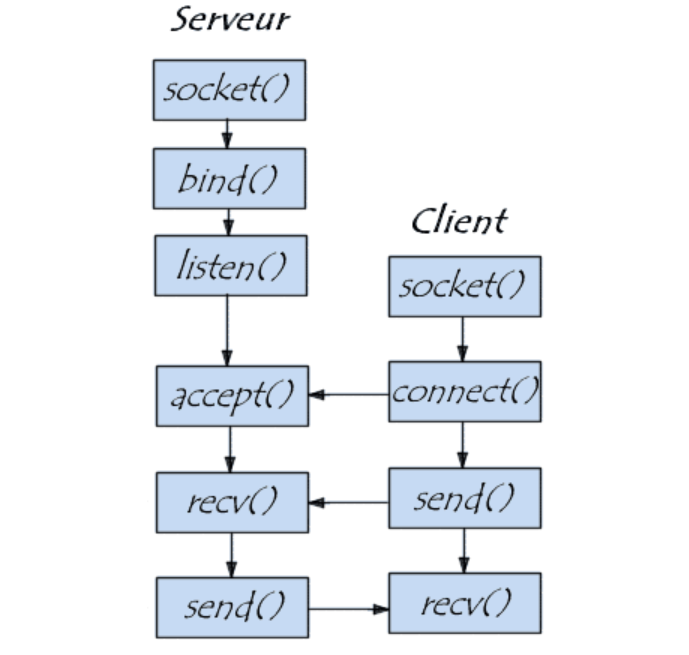


Figure 10 : Communication en mode connecté avec les sockets

* En mode non connecté

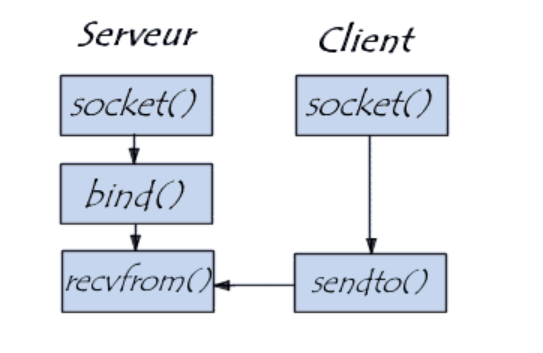
******

Figure 11: Communication en mode non connecté avec les sockets

Chapitre II

## I.Introduction

Il nous reste à mettre les concepts de JAVA en pratique sur une application clients/serveur de « ChatApp» pour cela nous avons choisis le langage JAVA avec l’environnement Eclipse, et nous avons utilisés « SWING » et « AWT » pour développer l’interface graphique, ainsi MySQL pour le but de crée une Base de données des clients, on utilise PHPMyAdmin sous Xamp Control Panel.

## II.Environnement de travail

### 1.Java

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 grande portabilité 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).

* **Caractéristiques du JAVA** :
* Orienté Objet.
* Distribué.
* Multithread.
* Robuste.
* Dynamique
* Portable.
* Haute performance.
* **Les composantes de java :**
* **Langage orienté objet JAVA.**
* **JVM (machine virtuelle Java) :** permette d’interpréter et d’exécuter le bytecode java
* **API (Application Programming Interface) :** un ensemble de classes standards (bibliothèque standard).
* **JRE (Java Runtime Environement)** : l’environnement d’exécution

Java désigne un ensemble d’outils permettant l’exécution de programmes

Java sur toutes les plates-formes supportées.

JRE est constitué de la JVM et de la Bibliothèque Standard (API).

* **Java Development Kit (JDK) :** le nouveau terme c’est SDK (Standard

Développent Kit) qui est l’environnement dans lequel le code Java est

Compilé pour être transformé en bytecode afin que la machine virtuelle Java (JVM) puisse l’interpréter JDK est composé de :

* Javac : le compilateur.
* Javadoc : le générateur de documentation.
* Jar : L’archiver.
* Jdb : Le débogueur.
* JRE : Environnement d’exécution java.

2.Eclipse

Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plate−forme modulaire pour permettre de réaliser des développements informatiques.



Figure 12 :Logiciel Eclipse

I.B.M. est à l'origine du développement d'Eclipse qui est d'ailleurs toujours le cœur de son outil Websphere Studio Workbench (WSW), lui-même à la base de la famille des derniers outils de développement en Java d'I.B.M. Tout le code d'Eclipse a été donné à la communauté par I.B.M afin de poursuivre son développement.

Eclipse utilise énormément le concept de modules nommés "plug−ins" dans son architecture. D’ailleurs, hormis le noyau de la plate−forme nommé "Runtime", tout le reste de la plate−forme est développé sous la forme de plug−ins. Ce concept permet de fournir un mécanisme pour l'extension de la plate−forme et ainsi fournir la possibilité à des tiers de développer des fonctionnalités qui ne sont pas fournies en standard par Eclipse.

Les principaux modules fournis en standard avec Eclipse concernent Java mais des modules sont en cours de développement pour d'autres langages notamment C++, Cobol, mais aussi pour d'autres aspects du développement (base de données, conception avec UML, ...). Ils sont tous développés en Java soit par le projet Eclipse soit par des tiers commerciaux ou en open source.

Les modules agissent sur des fichiers qui sont inclus dans l'espace de travail (Workspace). L'espace de travail regroupe les projets qui contiennent une arborescence de fichiers.

Bien que développé en Java, les performances à l'exécution d'Eclipse sont très bonnes car il n'utilise pas Swing pour l'interface homme−machine mais un toolkit particulier nommé SWT associé à la bibliothèque

JFace. SWT (Standard Widget Toolkit) est développé en Java par IBM en utilisant au maximum les composants natifs fournis par le système d'exploitation sous-jacent. JFace utilise SWT et propose une API pour faciliter le développement d'interfaces graphiques.

#### Les points forts d'Eclipse

Eclipse possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

Eclipse possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

* Une plate−forme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plug−ins
* Plusieurs versions d'un même plug−in peuvent cohabiter sur une même plate−forme.
* Un support multi langage grâce à des plug−ins dédiés : Cobol, C, PHP, C#, ...
* Support de plusieurs plateformes d'exécution : Windows, Linux, Mac OS X, ...
* Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT
* Les nombreuses fonctionnalités de développement proposées par le JDT (refactoring très puissant, complétion de code, nombreux assistants, ...)

Une ergonomie entièrement configurable qui propose selon les activités à réaliser différentes « Perspectives »

### 3.AWT et SWING

La première API pour développer des interfaces graphiques portables d'un système à un autre en Java est AWT. Cette API repose sur les composants graphiques du système sous-jacent ce qui lui assure de bonnes performances. Malheureusement, ces composants sont limités dans leur fonctionnalité car ils représentent le plus petit dénominateur commun des différents systèmes concernés. Pour pallier ce problème, Sun a proposé une nouvelle API, Swing. Swing fait partie de la bibliothèque Java Foundation Classes (JFC). C'est une API dont le but est similaire à celui de l'API AWT mais dont les modes de fonctionnement et d'utilisation sont complètement différents. Swing a été intégré au JDK depuis sa version 1.2. Cette bibliothèque existe séparément pour le JDK 1.1.

Swing utilise la même infrastructure de classes qu'AWT, ce qui permet de mélanger des composants Swing et AWT dans la même interface. Il est toutefois recommandé d'éviter de les utiliser simultanément car certains peuvent ne pas être restitués correctement. Les principales différences sont :

* Utilisation du double buffering qui améliore les rafraichissements.
* Utilisation d'un panneau de contenu (contentPane) pour insérer des composants.

Le double-buffering est le fait d'avoir un buffer où l'on dessine en mémoire et un buffer qui s'occupe de l'affichage. Principe est : dessiner sur le buffer mémoire (des images, des textes, des dessins...), ensuite, dessiner le buffer mémoire dans le buffer d'affichage (séparation entre le chargement et l’affichage).

Les composants Swing ne sont plus insérés directement au JFrame mais à l'objet

contentPane (de la classe JRootPane) qui lui est associé.

### 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 plateforme sur laquelle elle s'exécute.

#### **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 Connection 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).

##### ***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.

##### ***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.

### 5.UML

UML est apparue les depuis les années 1980 aussi le langage pour la description de la programmation par objets a fait ressortir l’étendue du travail conceptuel nécessaire ; définition des classes, de leurs relations, des attributs et méthodes, des interfaces etc.

Pour programmer une application, il ne convient pas de se lancer tête baissée dans l’écriture du code : il faut d’abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation. C’est cette démarche antérieure à l’écriture que l’on appelle modélisation ; son produit est un modèle. Les spécifications fournies par la maîtrise d’ouvrage en programmation impérative étaient souvent floues : les articulations conceptuelles (structures de données, algorithmes de traitement) s’exprimant dans le vocabulaire de l’informatique, le modèle devait souvent être élaboré par celle-ci. L’approche

Objet permet en principe à la maîtrise d’ouvrage de s’exprimer de façon précise selon un vocabulaire qui, tout en transcrivant les besoins du métier, pourra être immédiatement compris par les informaticiens. Concernant notre projet on a utilisé Sybase PowerAMC.

6.XAMPP Control Panel

**XAMPP** est un ensemble de logiciels permettant de mettre en place facilement un serveur Web et un serveur FTP. Il s’agit d’une distribution de logiciels libres (X Apache MySQL Perl PHP) offrant une bonne souplesse d’utilisation, réputée pour son installation simple et rapide. Ainsi, il est à la portée d’un grand nombre de personnes puisqu’il ne requiert pas de connaissances particulières et fonctionne, de plus, sur les systèmes d’exploitation les plus répandus.

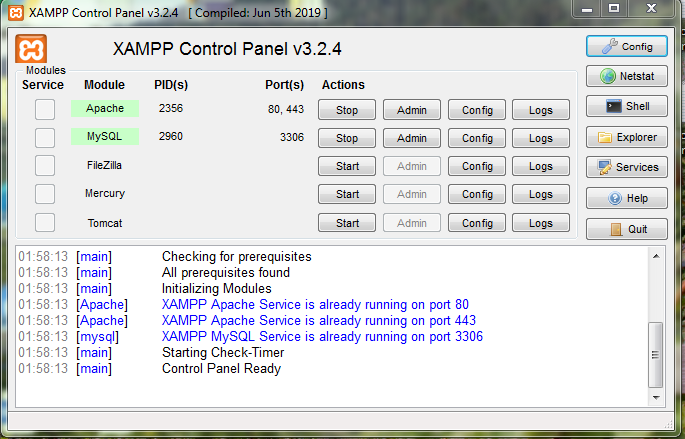
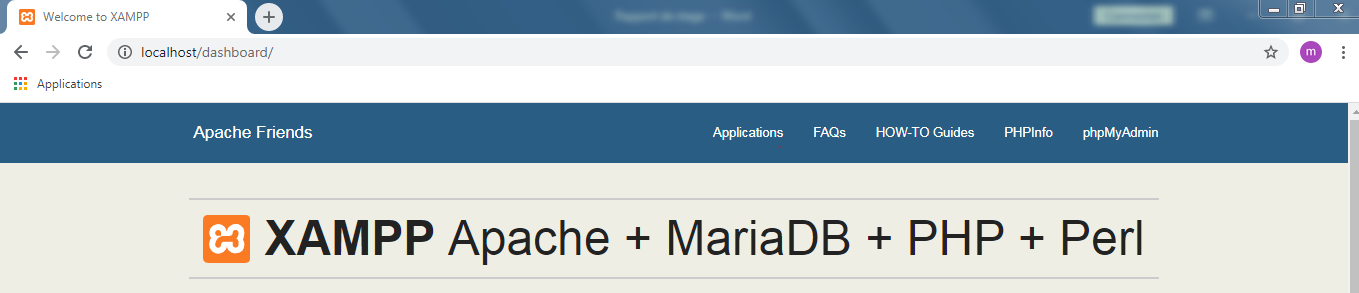


Figure 13 : XAMPP Control Panel v3.2.4

Cette « distribution » se chargera donc d’installer l’ensemble des outils dont vous pourriez avoir besoin lors de la création d’un site Web. Plus d’une dizaine d’utilitaires sont intégrés, comme MySQL, PHP, Perl ou encore phpMyAdmin. Il est distribué avec différentes bibliothèques logicielles qui élargissent la palette des services de façon notable : OpenSSL, Expat(parseur XML), PNG, SQLite, zlib, … ainsi que différents modules Perl et Tomcat, FileZilla Server.

***7.MySQL***

MySQL/SQL permet la communication entre les bases de données et les sites Web, ce qui permet aux utilisateurs de tester et de stocker les emails, les mots de passe et toute autre donnée. On clique sur le bouton Admin et aprés on attend qu’Apache et MySQL soient surlignés en vert sous votre section de module.



**Figure 14 : page d’accueil de XAMP**

Ensuite, on clique sur le bouton **phpMyAdmin** dans le menu de navigation supérieur. et la page phpMyAdmin est lancer par défaut comme suit . Apres on a commencé à configurer vos bases de données MySQL. L’écran de connexion phpMyAdmin s’ouvrira dans une nouvelle fenêtre du Navigateur. Par défaut, le nom d’utilisateur administrateur sera **root**, et vous pouvez laisser le champ mot de passe vide.

A partir de là, on a créé de nouvelles bases de données MySQL nommé **base1** ,pour les clients.

## IV.Présentation de l’application

Notre projet consiste à réaliser une application Client-Serveur par JAVA SWING, AWT, Networking. Dans un réseau local ne demande pas une connexion internet le but de l’application et d’échanger des messages en temps réel au sein de ce réseau.

## V.Modélisation UML

### 1. Diagramme de classe

Le diagramme de classes est généralement considéré comme le plus important dans un développement orienté objet. Il représente l’architecture conceptuelle du système : il décrit les classes que le système utilise, ainsi que leurs liens, que ceux-ci représentent un emboîtage conceptuel (héritage) ou une relation organique (agrégation). Pour notre projet on fait le diagramme suivant :

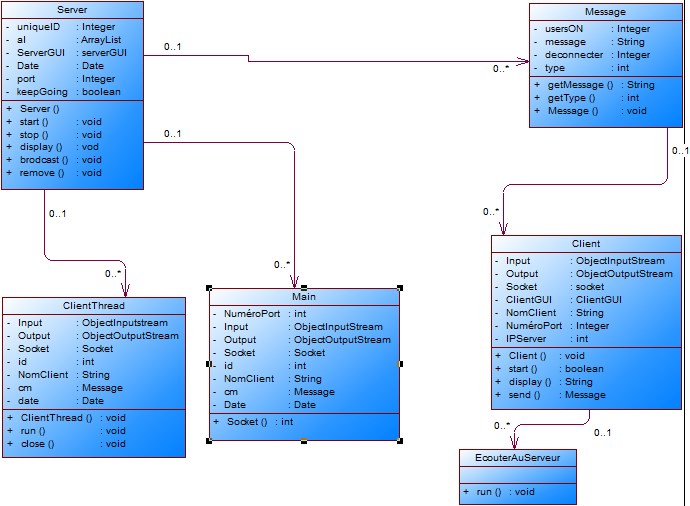


Figure 15 : Diagramme de classe

### 2.Diagramme de séquence

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur. Il indique les objets que l’acteur va manipuler et les opérations qui font passer d’un objet à l’autre. On peut représenter les mêmes opérations par un diagramme de communication graphe dont les nœuds sont des objets et les arcs (numérotés selon la chronologie) les échanges entre objets. En fait, diagramme de séquence et diagramme de communication sont deux vues différentes mais logiquement équivalentes (on peut construire l’une à partir de l’autre) d’une même chronologie. Ce sont des diagrammes d’interaction. Pour notre projet on fait le diagramme suivant :

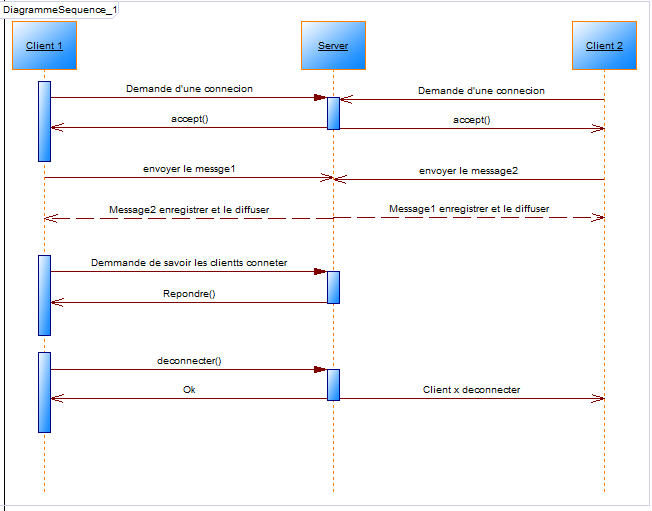


Figure 16 : Diagramme de séquence

## V.Les interfaces de l’application

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 compréhensibilité, etc.

Dans ce qui suit une présentation des captures d’écrans des plus importantes interfaces de notre application.

### 1.Coté Serveur

Afin d’échanger des messages en temps réel il faut que l’administrateur de serveur donne a tous ses clients le droit de se connecter sur le serveur via la Button Commencer.

L’interface d’accueil de serveur Contient 3 interfaces Chat, Evenment,Clients et aussi le numéro de port utilisé par le serveur.

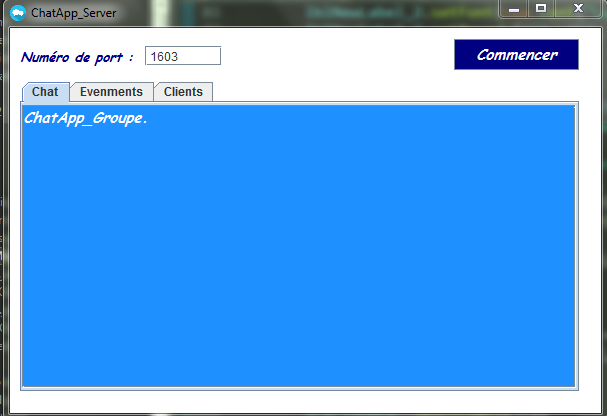
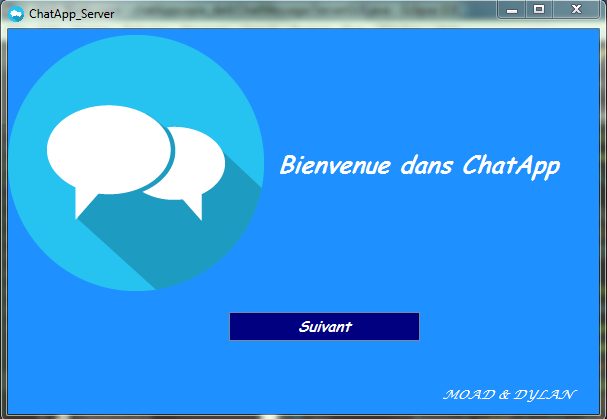


Figure 17 : Les interfaces d’accueil de serveur

Après le serveur reste a écouter sur le port 1603 des des messages venus de n’importe quelle client qui se trouve dans sa base de donnée.

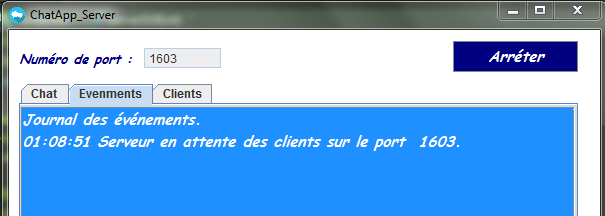


Figure 18 : L’interface d'évènement du serveur

Pour couper toute connexion à travers le serveur l’administrateur doit utiliser la Button Arrêter

* **Chat**

Cette interface sert à afficher les messages transmet entre les clients de serveur avec une la date précise de chaque message.

* **Evènement**

Chaque Action est faite par un Client quiconque est directement afficher dans cette interface avec la date précise.

* **Clients**

Chaque Client est connecté sur le serveur est directement stocker dans une liste des clients

### 2.Coté client

Seuls les Clients qui sont dans la base de données de Serveur qui ont le droit d’utilisé l’application.

Afin d’appliqué cette règle on a créé une base de données pour les clients a travers PHPMyAdmin dans XAMPP Control Panel comme suit



Figure 19 : MySQL phpMyAdmin

Et on a inséré 3 clients « MOAD » et « DYLAN » et « User1 » dans une base de donnés base1 et chaque Client possède un Nom et Mot de passe.



Figure 20 : L’interface d’insertion des utilisateurs

#### **Interface d’authentification :**

Cette première capture présente l’interface d’authentification dans laquelle on doit saisir le nom d’utilisateur et entrer le mot de passe pour commencer à utiliser l’application. Cette étape met en valeur l’aspect de sécurité, c.-à-d. nous vérifions la disponibilité du compte utilisateur dans la base de données de serveur.

En gros cette fenêtre nous permet d’entrer à l’application, donc l’utilisateur doit entrer son nom et le mot de passe convenables ;



Figure 21 : L’interface d’authentification des Clients

Aussi si le mot de passe ou le nom d’utilisateur est incorrecte, il nous affiche un message d’erreur, sous forme d’un pop-up, qui montre qu’il y’a quelque chose qui ne va pas, comme ci-dessous :

- Le message est comme ceci : « Login et Mot de passe est Invalide ».

- On clique sur le bouton « OK » pour recommencer à nouveau.

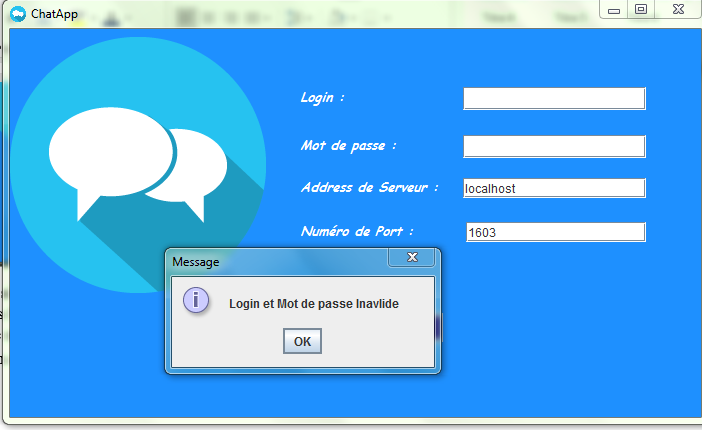


Figure 22 : L’interface de vérification du compte utilisateur

#### **Interface de la page d’accueil :**

Cette capture représente la page d’accueil de notre application client.

Elle se compose tabbedPane « Chat » pour l’affichage des messages transmise et une TextFieled pour l’écriture de message et les 3 boutons principaux c.-à-d. 3 fonctions différentes :



L’interface d’accueil de l’utilisateur

* Button « ***Se deconnecter*** » sert à déconnecter d’une communication via ce serveur comme suit :
* Button « ***Utilisateurs*** » set a affiché les utilisateurs qui ont connectées sur le serveur
* Button « ***Routeur*** » pour a la page d’authentification

## VI. Fonctionnement de l’application

Pour tester le fonctionnement de l’application on prendre deux et il y’ la possibilité de prendre plus utilisateurs « MOAD » et « DYLAN » appartient aux bases de données de serveur.

Alors pour l’utilisateurs « MOAD » est venu de se connecter sur le serveur

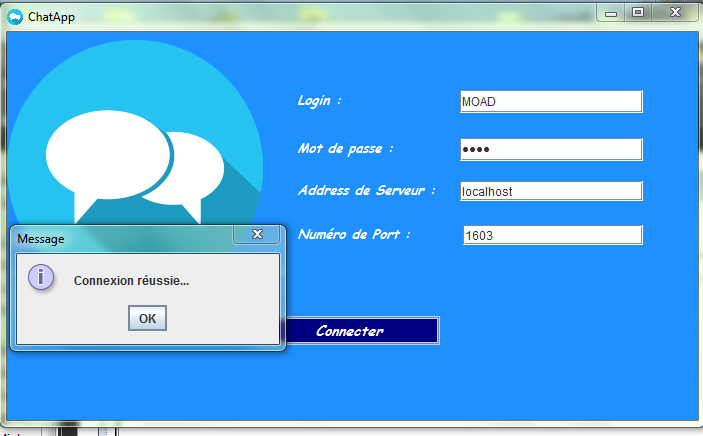


Figure 23 : Connexion du compte utilisateur MOAD

Alors pour l’utilisateurs « DYLAN » est venu de se connecter sur le serveur

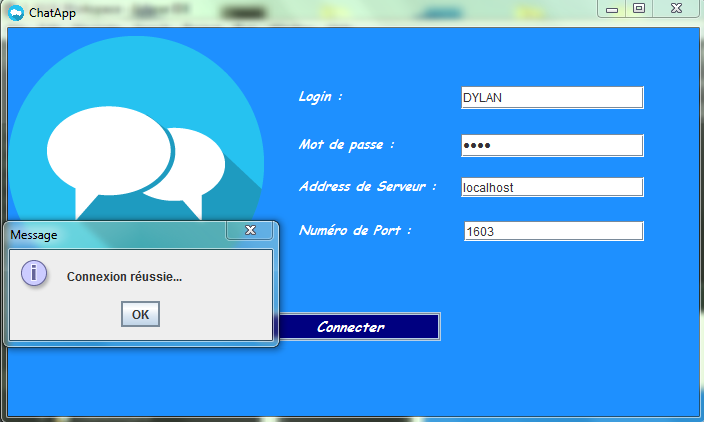


Figure 24 : Connexion de l'utilisateur « DYLAN »

\*

Pour le Serveur

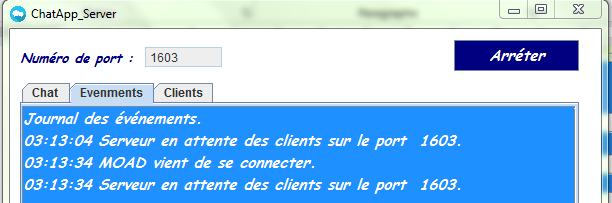
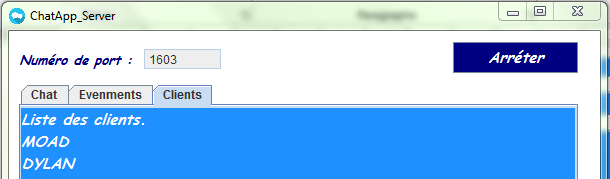


Figure 25 : L'interface de l'affichage des clients connectés

Figure 26 : L’interface de l’évènement sur le serveur

Donc les deux utilisateurs sont maintenant connectés sur le Serveur ,

Vérification la transmission des messages entre les deux utilisateurs

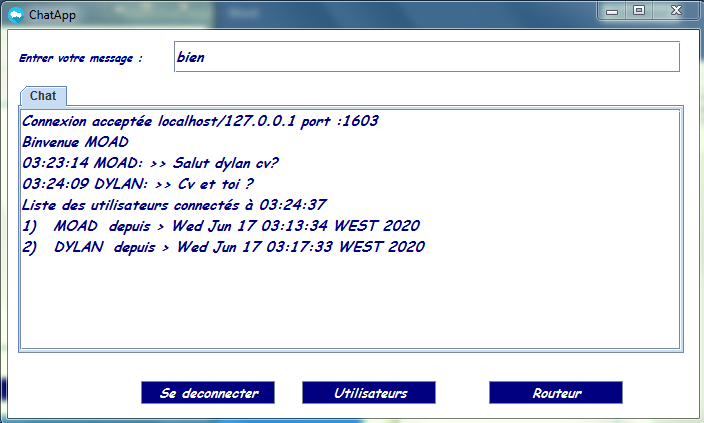


Figure 27 : les interface de connexion de l'utilisateur Moad

Figure 28 : L'interface de connexion de l'utilisateur Dylan

Donc on voit bien que les deux utilisateurs peuvent communiquer entre eux et que la Button Utilisateurs afficher une liste des utilisateurs qui ont connectées.

**Pour le serveur**

On remarque que les deux messages sont enregistrés dans l’interfaces « Chat » de Serveur

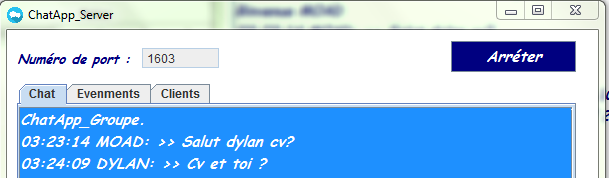
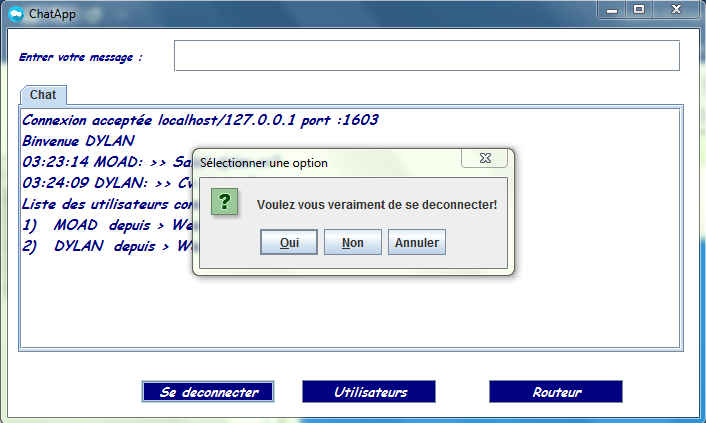


Figure 29 : L’interface d’affichage des messages dans le serveur

Et en fin si l’utilisateurs veux de se déconnecter juste on clique sur le Button « Se deconnecter »

Et il nous a afficher une fenêtre Sélection une Option pour la confirmation de la déconnection ou non



L’interface pour la confirmation des déconnections des utilisateurs

## **L’utilité de l’application**

Alors notre application « ChatApp » peut être utilisable dans n’importe quelle organisme ou entreprise a une réseau local privé ,l’administrateur du réseau peut simplement installer et administrer l’application dans le réseau propre d’un entreprise ou organisme sans besoin d’un connexion internet et ce stade l’entreprise sera réduire le coût de la connexion Internet et aussi sera la possibilité de communiqué au sein de sa propre réseau de manière sécurisable et faible surtout dans des période des confinements ou on a beaucoup des Cyber-attaques contre les organismes.

## VII.Conclusion

Ce projet nous a donné la chance de découvrir les différentes architectures distribuées, en particulier l’architecture de type Clients/Serveur et de se familiariser aussi avec l’outil Java et en particulier avec JAVA Swing et Awt et JAVA Networking.

Nous avons appris comment utiliser l’architecture Clients /Serveur avec la programmation réseau des Socket qui est dans JAVA et aussi Nous avons apprendre la méthode de développer une application de messagerie en temps réel i.e. la messagerie instantanée dans un réseau local avec l’intégration d’une base de données MySQL.

Bibliographie

**[6] http**://www.JDBC.com juin 2013

**[7]** Structured Query Language-Wikipédia.htm.

**[8]** Développons en Java-JDBC (Java DataBase Connectivity).htm