



UNIVERSITÉ ABDELMALEK ESSAÂDI

Faculté des Sciences

Tétouan



Master spécialisé d'informatique

M2I/MQL

Projet de Fin de module

**Développement d'un système
de planification de tâches
distribuées**

Pr.	EL HIBAOUI Abdelaziz	Département d'Informatique	Examineur
-----	----------------------	----------------------------	-----------

Réalisé par :

ELFRAANI Yousra (MQL)

BENTHAMI EL HAMDOUCHI Mouad(M2I)

Soutenu le :

20/02/2023

Année universitaire 2022-2023

Table des matières

Introduction general.....	5
Chapitre 1 : Contexte générale	6
Problematique	7
Objectif	7
Chapitre 2 : Planification et conception.....	8
planification	9
Archetector clients-serveur-esclaves	9
Diagramme de gantt.....	9
Conception.....	10
UML (Unified Modeling Language)	10
Diagramme de cas d'utilisation	11
Les acteurs et leurs rôles.....	12
Diagramme de séquences	13
Diagramme de classe	14
Chapitre 3 : Réalisation :	15
Les interfaces principaux	16
Conclusion:	19

Table des figures

Figure 1 : Architecture Clients-Serveur-Esclaves.	9
Figure 2 : Diagramme de Gant.	9
Figure 3 : Logotype de UML	10
Figure 4 : Visual paradigm	10
Figure 5 : Diagramme de cas d'utilisation client-serveur.....	11
Figure 6 : Diagramme de cas d'utilisation serveur-esclave.....	11
Figure 7 : Diagramme de séquence.....	13
Figure 8 : Diagramme de Classe.	14
Figure 9 :Fichier de configuration Serveur.....	16
Figure 10 :La liste des taches.....	16
Figure 11:Tache 1.	16
Figure 12 :Tache 2	17
Figure 13: Tache 3	17
Figure 14 :Tache 4	17
Figure 15 :Fichier de configuration Client.	18
Figure 16 :Fichier de configuration Slaves	18

Introduction générale

La planification de tâches distribuées est devenue un sujet incontournable dans les environnements informatiques actuels, en raison de la nécessité de maximiser l'efficacité et la rapidité de traitement pour répondre aux besoins en constante évolution des entreprises et des organisations. Ce système permet de centraliser la gestion des demandes de tâches en provenance de différents clients, et de les distribuer de manière efficace à plusieurs ressources pour traitement. Cette approche permet une utilisation optimale des ressources et une vitesse de traitement accélérée, offrant ainsi des avantages considérables pour les entreprises et les organisations.

Le système de planification de tâches distribuées implique plusieurs parties importantes, telles que le serveur central, les clients, les esclaves et les algorithmes de planification. Ce rapport détaillera les différentes étapes du développement du système, y compris la conception et la réalisation.

Chapitre I :

« Contexte générale »

➤ Problématique :

La problématique liée au sujet de la planification de tâches distribuées peut être décrite comme suit :

- ✚ Difficulté pour les clients de soumettre des tâches à exécuter rapidement
- ✚ Difficulté pour le serveur de gérer de manière efficace un grand nombre de tâches provenant de clients différents.
- ✚ Difficulté de gestion des tâches pour leur envoi aux esclaves correspondants par le serveur.
- ✚ Difficulté à assurer la fiabilité de l'envoi des résultats aux clients correspondants pour le serveur.

➤ Objectif :

L'objectif de ce projet est de créer un système de planification de tâches distribuées pour aider plusieurs clients de se connecter à un serveur et de soumettre des tâches à exécuter de manière efficace et rapide. Le serveur sera chargé de la gestion des tâches, de leur distribution aux différents esclaves, et de la réception et de l'envoi des résultats aux clients correspondants. En utilisant des algorithmes de planification avancés, le système minimisera le temps nécessaire pour soumettre et exécuter les tâches et garantir la fiabilité de l'envoi des résultats aux clients correspondants.

Chapitre II :

« Planification et Conception »

II.1-Planification :

➤ Architecture Clients-Serveur-Esclaves :

L'architecture clients-serveur-esclaves est un modèle informatique dans lequel un ordinateur client transmet une requête à un ordinateur pour accéder à des ressources ou des services.

Une particularité est que le serveur reçoit les requêtes des clients et envoie des tâches aux serveurs secondaires (ou esclaves) pour les traiter. Les esclaves travaillent avec le serveur principal pour gérer la charge de travail en distribuant les tâches entre eux.

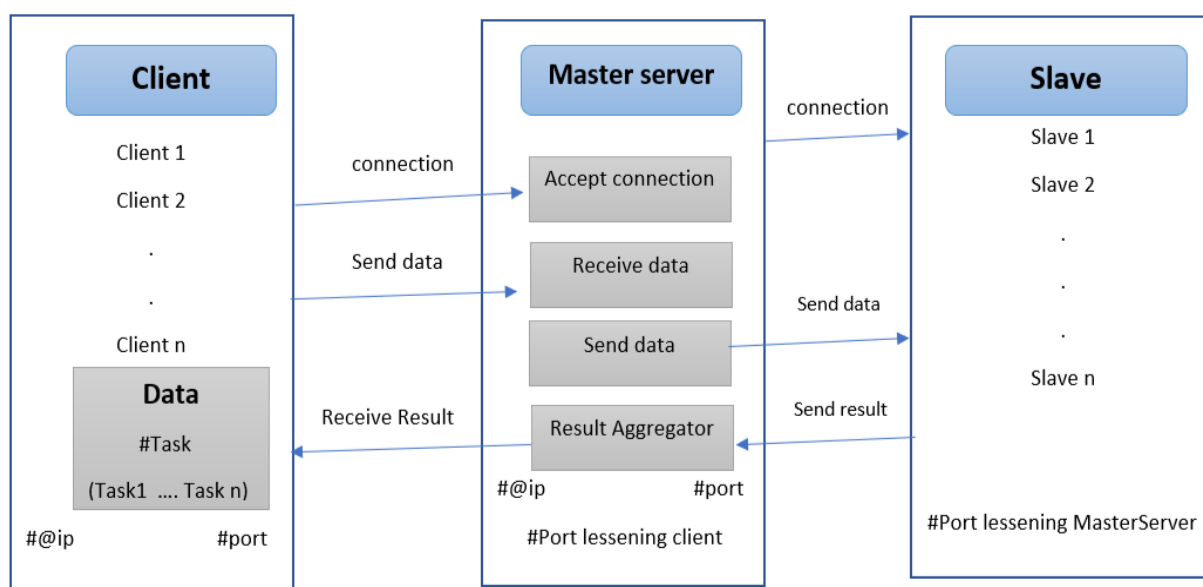


Figure1 : Architecture Clients-Serveur-Esclaves.

➤ Diagramme de Gantt :

Dans un diagramme de GANTT chaque tâche est représentée par une ligne, tandis que les colonnes représentent les jours, semaines ou mois du calendrier selon la durée du projet.

	jour 1	jour 2	jour 3	jour 4	jour 5	jour 6	jour 7	jour 8	jour 9	jour 10	jour 11	jour 12	jour 13	jour 14	jour 15	jour 16
Architecture client/serveur																
Conception UML																
Reaaliation d application																
Tests d'acceptation																
Realisation du rapport																

Figure2 : Diagramme de Gant.

II.2-Conception :

➤ 2.a UML (Unified Modeling Language)

- **Définition :**

L'UML se définit comme un langage de modélisation graphique et textuel. Il est destiné à comprendre et décrire des besoins, spécifier et documenter les systèmes, et sert aussi à esquisser des architectures logicielles, concevoir des solutions et communiquer des points de L'UML unifie à la fois les notations et les concepts orientés objet. Il ne s'agit pas d'une simple notation graphique, car les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage.



Figure3 : Logotype de UML

- **Outil de modélisation :**

Visual Paradigme (VP-UML) est un outil UML CASE prenant en charge UML 2, SysML et la notation de modélisation des processus métiers (BPMN) du groupe de gestion d'objets (OMG). En plus de la prise en charge de la modélisation, il fournit des capacités de génération de rapports et d'ingénierie de code, y compris la génération de code.



Figure4 : Visual paradigm

➤ 2.c Diagramme de cas d'utilisation :

Les diagrammes de cas d'utilisation sont des digrammes UML qui représentent les services les plus importants rendus par un système.

🚦 Client-Serveur :

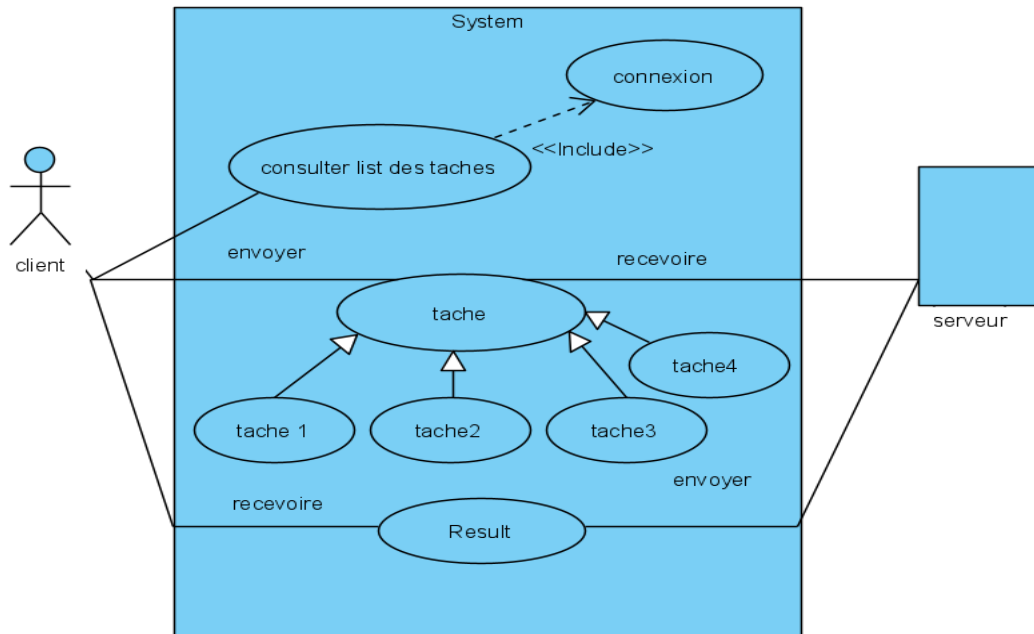


Figure 5 : Diagramme de cas d'utilisation « client-serveur »

🚦 Serveur-Esclave :

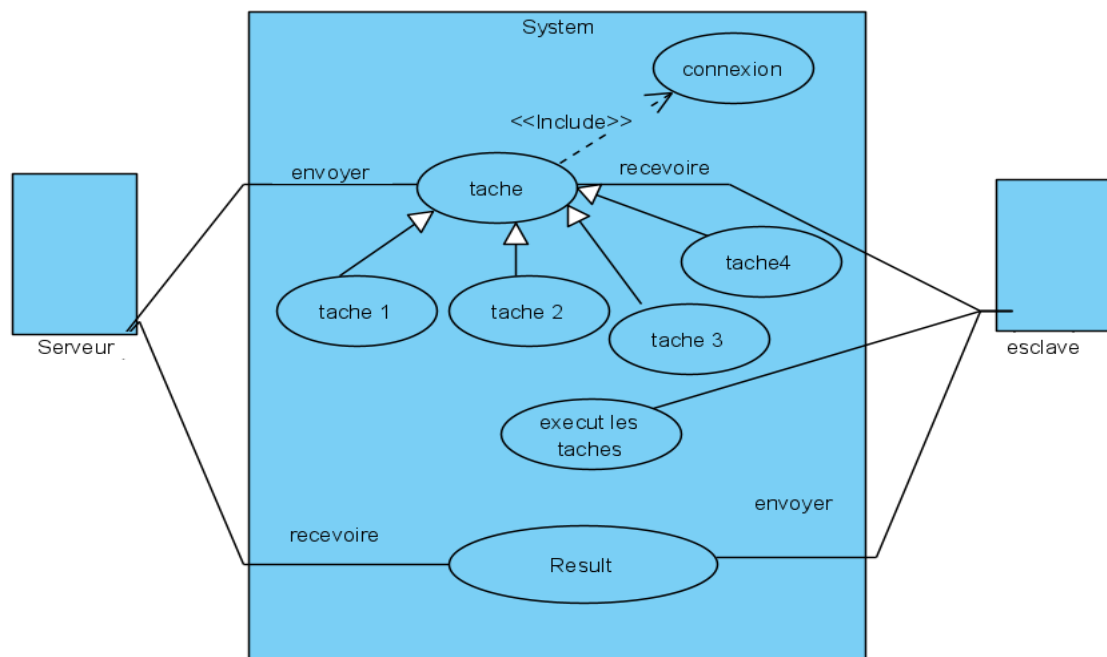


Figure 6 : Diagramme de cas d'utilisation « serveur-esclave »

➤ 2.c Les acteurs et leurs rôles :

Dans cette section, nous allons décrire les différents acteurs impliqués dans le système informatique. Après une analyse approfondie, nous avons identifié trois acteurs :

- **Client** : Cet acteur est capable d'envoyer une ou plusieurs tâches au système et de recevoir les résultats de chaque tâche.
- **Serveur** : Ce serveur principal est responsable de recevoir les tâches des clients et de les distribuer aux esclaves. Il est également en charge de recevoir les résultats des tâches traitées par les esclaves et de les renvoyer aux clients.
- **Esclave** : Les esclaves, sont chargés d'exécuter les tâches reçues du serveur principal et de renvoyer les résultats au serveur principal.

<i>Acteur</i>	<i>Rôles</i>
Client	<ul style="list-style-type: none">▪ Envoyer une tâche ou plus▪ Recevoir le résultat de chaque tâche
Serveur	<ul style="list-style-type: none">▪ Accepter la connexion des clients.▪ Recevoir les tâches des clients.▪ Stocker les tâches dans une file d'attente.▪ Distribuer les tâches de manière séquentielle, aux esclaves.▪ Récupérer les résultats des tâches traitées▪ Renvoyer les résultats des tâches traitées aux clients correspondants
Esclave	<ul style="list-style-type: none">▪ Accepter la connexion de serveur.▪ Recevoir les tâches▪ Stocker les tâches reçues dans une file d'attente.▪ Traiter les tâches de manière séquentielle.▪ Envoyer les résultats au serveur.

➤ 2.d Diagramme de séquences :

Les diagrammes de séquences permettent de décrire comment les éléments du système interagissent entre eux et avec les acteurs, ses principales informations sont les messages échangés entre les lignes de vie, présentés dans un ordre chronologique.

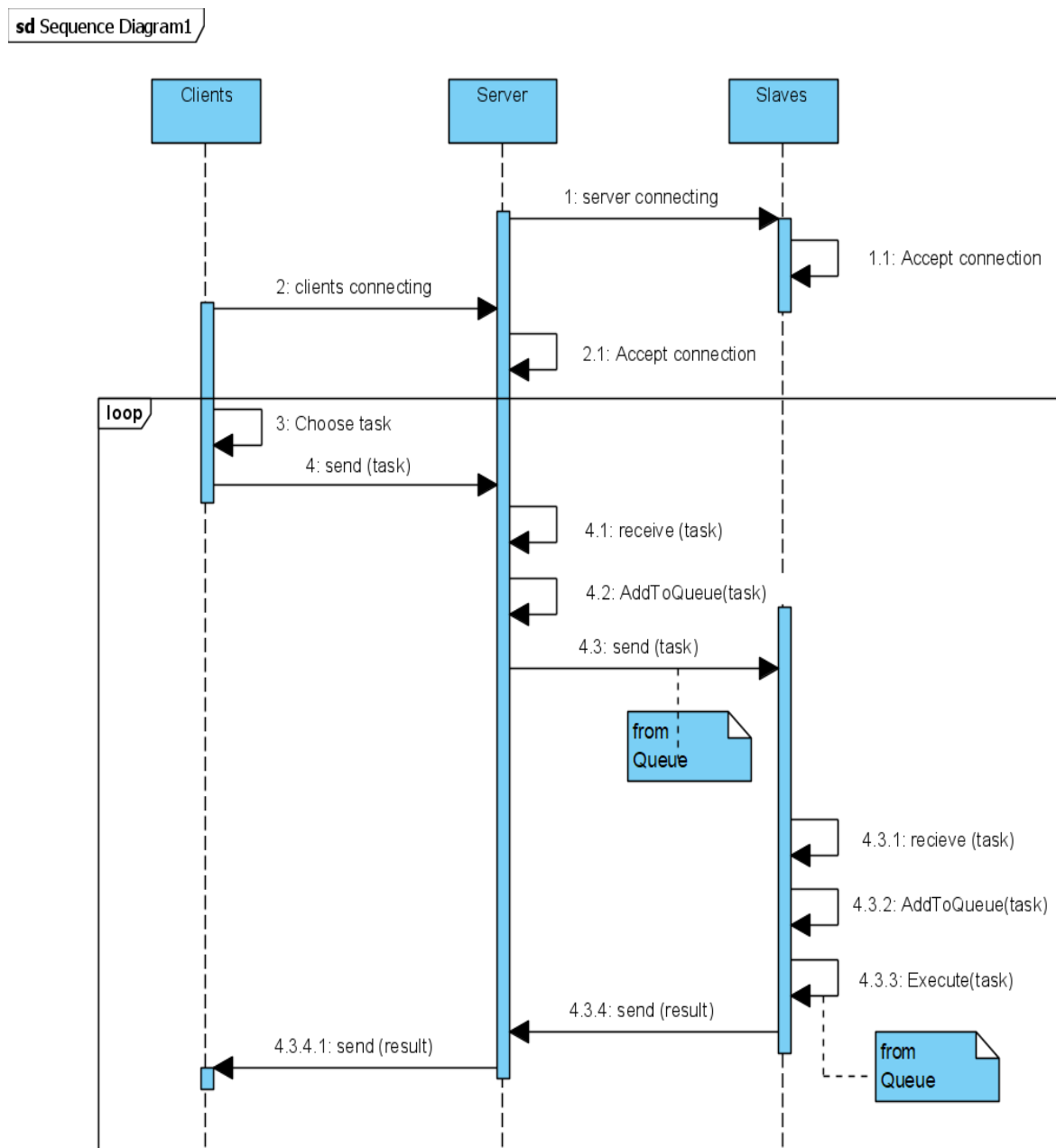


Figure 7 : Diagramme de séquence.

➤ 2.E Diagramme de classe

Le diagramme de classe est le point central dans le développement orienté objet, elle représente la structure de système sous forme de classes et de relations entre classes. Ces classes constituent la base pour la génération de code et pour la génération des schémas de bases de données

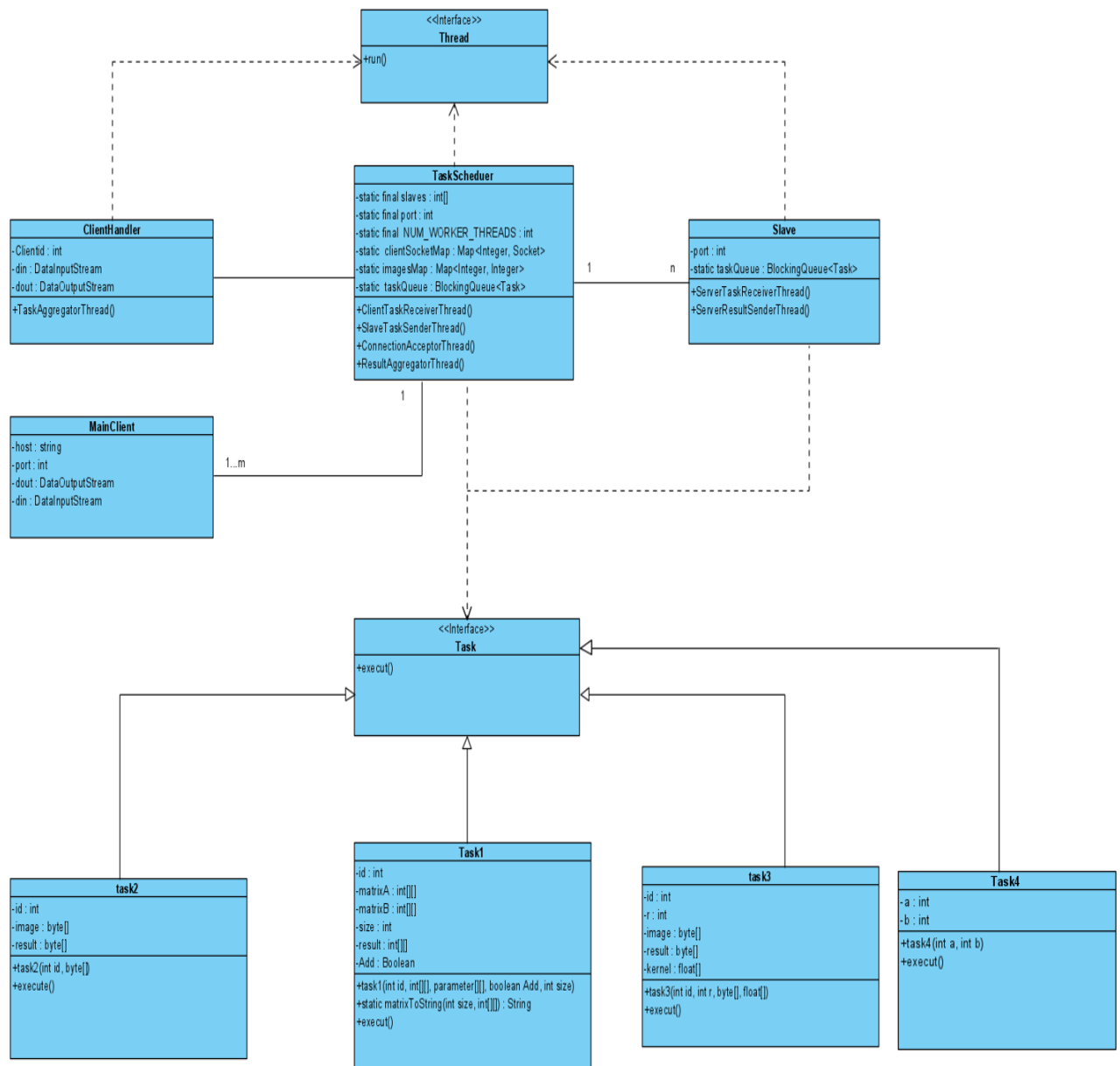


Figure 8 : Diagramme de classe.

Chapitre III :

« Réalisation »

III.1-Les interfaces principaux :

➤ 1.a Clients :

- Le client établit une connexion avec le serveur en utilisant les informations du serveur principal contenues dans le fichier config.txt.

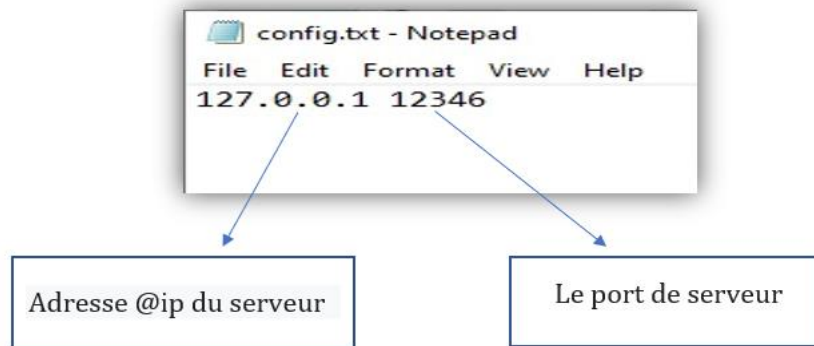


Figure 9 : Fichier de configuration Serveur.

- Après avoir établi la connexion, le client peut consulter la liste des tâches à exécuter.

```
Connected to server: /127.0.0.1:12346
ready
chose a task :
task1 : calculate the sum from a to b
task2 : calculate the sum or difference of two matrices
task3 : apply a black and white filter to an image
task4 : apply a convolution product to an image using a kernel
To exit type end
```


Figure 10 : la liste des taches.

- Ensuite, le système fournit les informations nécessaires en fonction de la tâche sélectionnée par le client. Une fois la tâche terminée, le client recevra les résultats et pourra ensuite décider s'il souhaite effectuer une autre tâche ou mettre fin à la session.

✚ **Tache 1 :** la somme du premier nombre jusqu'à le deuxième nombre.

```
task1
enter a 1
enter b 10
Result : 55
chose a task :
```

Figure 11 : tache 1.

 **Tache 2** : L'addition ou une soustraction de deux matrices.

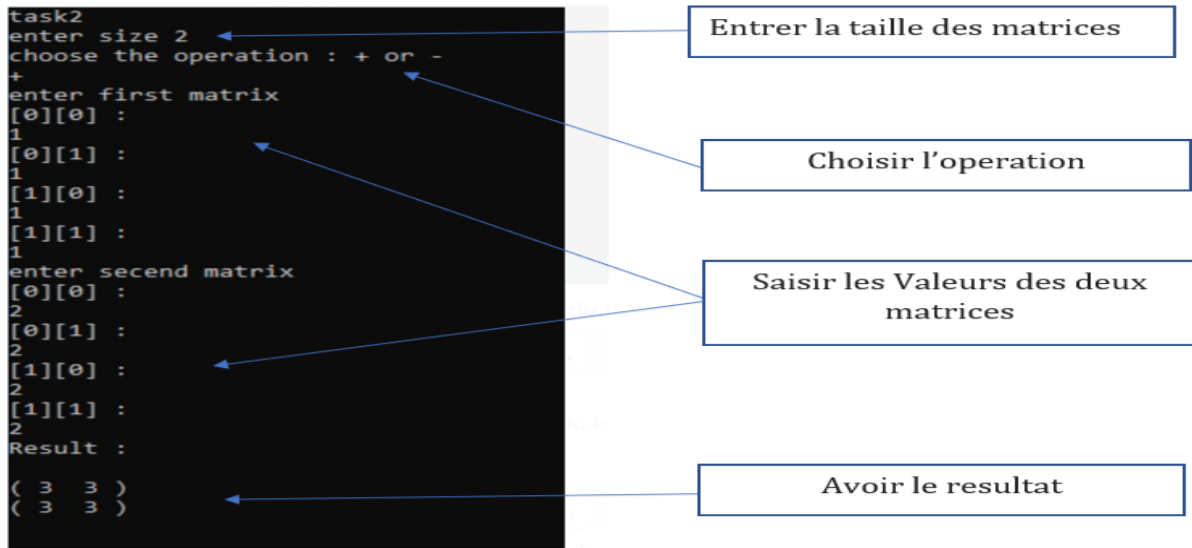


Figure 12 : tache 2.



 **Tache 3** :L'application du filtre Blanc-Noir.



Figure 13 : tache 3.

 **Tache 4** : L'application du filtre produit de convolution.

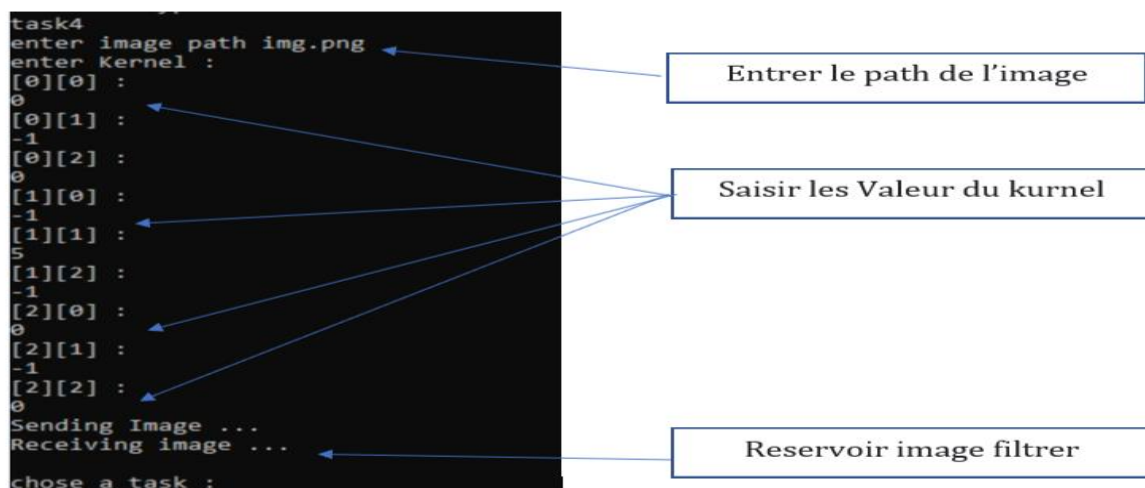


Figure 14 : tache 4.

➤ 1.b Serveur :

- Le serveur principal utilise un fichier de configuration pour récupérer le port client, ainsi qu'un autre fichier contenant les adresses IP et les ports des esclaves connectés.

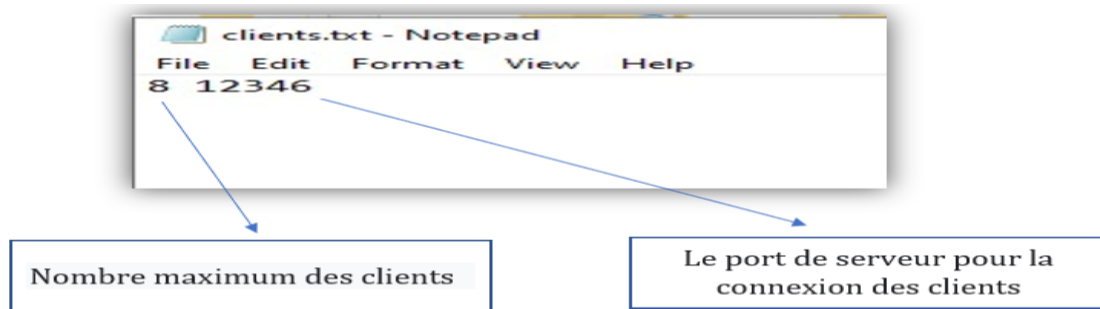


Figure 15 : Fichier de configuration Client.

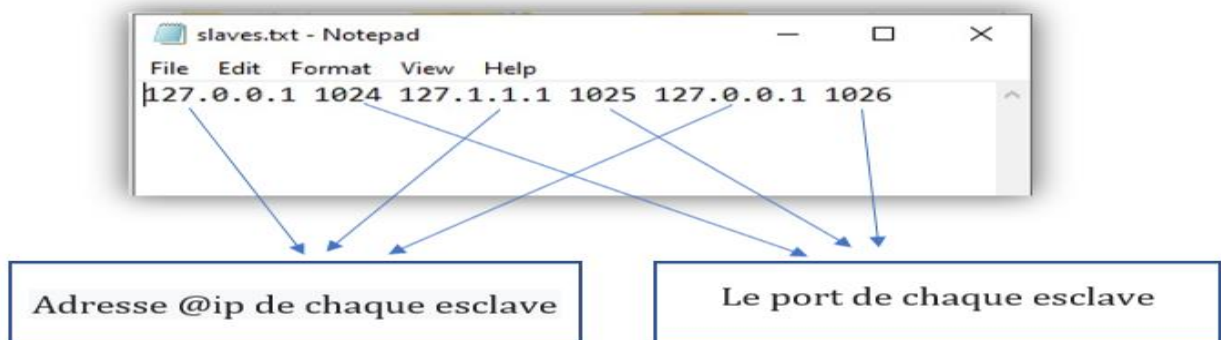


Figure 16 : Fichier de configuration Slaves.

- Lorsqu'il reçoit une tâche d'un client, le serveur l'ajoute à une liste d'attente et les traite successivement. Si la tâche n'est pas un produit de convolution, le serveur l'envoie à un esclave pour son exécution. En revanche, si la tâche est un produit de convolution, le serveur divise l'image en un nombre de morceaux égal au nombre d'esclaves et envoie chaque morceau à un esclave avec le noyau correspondant pour son exécution.
- Le serveur reçoit les résultats des tâches des esclaves et les redirige vers les clients en fonction de leur ID. Si la tâche est un produit de convolution, le serveur rassemble les résultats des différents morceaux traités par les esclaves, puis les redirige vers le client correspondant.

•

➤ 1.c Esclave :

- L'esclave introduit dans les arguments de ligne de commande un paramètre représente le port de connexion du serveur.
- Lorsqu'il reçoit la tâche du serveur, l'esclave les ajoute à une liste d'attente, puis les traite successivement. Une fois la tâche exécutée, l'esclave renvoie le résultat au serveur.

Conclusion

Le développement de notre application de planification de tâches distribuées nous a permis de mettre en pratique nos connaissances en langage Java, notamment en ce qui concerne les threads et les sockets. Notre objectif était de créer une application permettant de faciliter le travail des clients en centralisant la gestion des tâches distribuées.

Actuellement, Notre application permet à des clients de se connecter à un serveur pour lui soumettre des tâches qui seront exécutées par des esclaves. Nous sommes fiers d'avoir réussi à implémenter cette fonctionnalité qui permet de centraliser la gestion des tâches distribuées et de faciliter la réalisation des tâches pour les clients.

Nous avons trouvé ce projet passionnant car il nous a permis de nous familiariser avec les outils et les logiciels nécessaires pour atteindre notre objectif. Nous sommes convaincus que ce projet nous a permis d'acquérir de nouvelles compétences en matière de développement d'applications distribuées en Java.