

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique



Université des Sciences et de la Technologie Houari Boumediène

Faculté d'Informatique

Master en Systèmes Informatiques intelligents

Module : Technologie des agents

Rapport de projet de TP

Réalisé par :

AIT AMARA Mohamed, 181831072170

HAMMAL Ayoub, 181831048403

Année universitaire : 2021 / 2022

Plan du rapport

Partie 1 : Système expert	2
1. Inférence et chaînage avant	2
2. Implémentation du programme	2
2.1. Stratégie de résolution de conflits	3
2.2. Le système expert commercial proposé	3
3. Interface	4
3.1. L'interface JavaFX	4
3.2. L'interface Web	6
Partie 2 : Système Multi-Agents	7
1. Conception	7
2. Les interactions	8
3. Interface	9
Conclusion	12

Partie 1 : Système expert

1. Inférence et chaînage avant

Les systèmes experts sont des outils d'intelligence symbolique pour la résolution de problèmes d'un domaine spécifique. Avec l'aide d'un cognitif, les connaissances de l'expert du domaine sont retranscrites sous forme d'une base de règles ou base de connaissance. A l'instanciation des variables, le système cherche une séquence de règles applicables qui mène au but, soit par chaînage avant ou chaînage arrière.

Nous nous intéressons dans ce travail à la méthode de chaînage avant, qui consiste en l'algorithme suivant :

Étape 1 : Initialisation des variables du système expert

Étape 2 : Création de *l'ensemble de conflits* qui contient l'ensemble des règles applicables pour cette instanciation des variables.

Étape 3 : Si l'ensemble de conflits n'est pas vide, sélection d'une règle parmi l'ensemble de conflits à l'aide d'une stratégie de résolution de conflits.

Étape 4 : Application de la règle choisie et la retirer de la base de connaissance ou la marquer comme déjà utilisée.

Étape 5 : Répéter l'algorithme depuis l'étape 2 jusqu'à l'obtention d'une valeur pour la variable but ou bien jusqu'à ce qu'aucune règle ne puisse être appliquée.

À la fin de l'exécution de l'algorithme, nous obtenons un ensemble de valeurs pour les variables du système.

2. Implémentation du programme

Afin de concevoir le moteur d'inférence implémentant la stratégie de chaînage avant, nous devons d'abord concevoir les composants nécessaires pour la représentation de la base de connaissance. La figure 1 représente le diagramme de classe d'une règle de la base de connaissance. Une liste de règles représentant la base de connaissance est passée au moteur d'inférence, ainsi qu'une liste de valeurs initiales pour les variables - la mémoire de travail. Le moteur d'inférence met à jour les valeurs des variables au fur et à mesure des itérations.

À chaque itération de l'algorithme, nous pouvons récupérer le contenu de l'ensemble de conflits, et la règle sélectionnée depuis le moteur d'inférence.

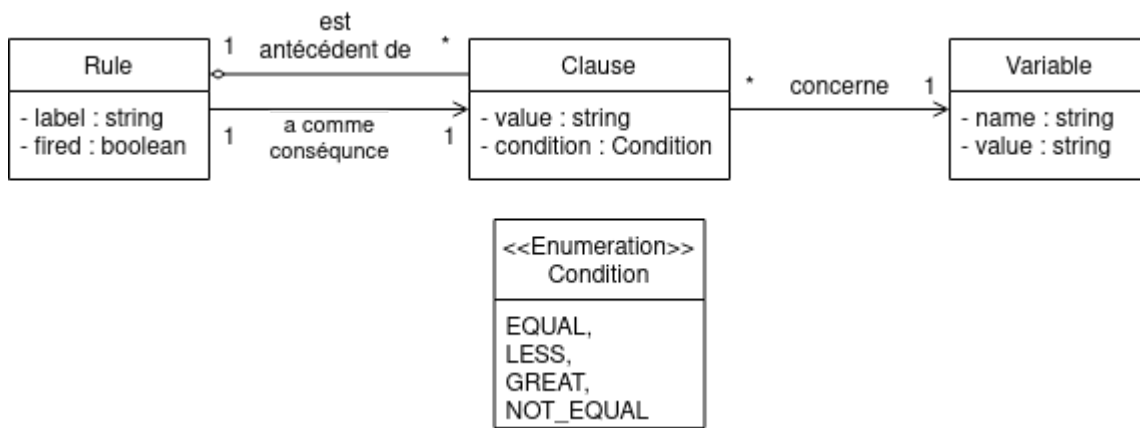


Figure 1 : Diagramme de classe partiel

2.1. Stratégie de résolution de conflits

La stratégie employée pour la résolution de conflits est la suivante : D'abord la règle avec le plus grand nombre de clauses antécédentes est sélectionnée. Si plusieurs règles sont sélectionnées en même temps, celle la plus bas dans la liste des règles est considérée comme la plus profonde dans l'arbre de déduction et elle est sélectionnée. Donc l'ordre des règles importe dans le processus de sélection.

2.2. Le système expert commercial proposé

Nous proposons, en plus du système expert par défaut de véhicules, un autre système expert commercial concernant la vente de laptop. Les règles de ce système expert sont résumées dans le tableau ci-dessous. Nous pouvons remarquer aussi que les deux premières règles appellent les 4 règles suivantes. Les bases sont stockées sous format JSON et chargées par le programme au besoin.

Label	Clauses Antécédentes	Clause Conséquence
chromebook	fonction = office	office categorie = chromebook
ultrabook	fonction = office AND autonomie = haute	office categorie = ultrabook
asus chromebook	office categorie = chromebook AND prix = pas cher	laptop = asus chromebook
samsung chromebook	office categorie = chromebook	laptop = samsung chromebook
hp probook	office categorie = ultrabook AND prix = pas cher	laptop = hp probook

dell xps 13	office categorie = ultrabook	laptop = dell xps 13
razer blade	fonction = jeux AND poids = leger	laptop = razer blade
acer nitro 5	fonction = jeux AND prix = pas cher	laptop = acer nitro 5
lenovo legion 5	fonction = jeux	laptop = lenovo legion 5
apple macbook	fonction = pro AND utilisation pro = multimedia	laptop = apple macbook
lenovo thinkpad	fonction = pro AND utilisation pro = workstation	laptop = lenovo thinkpad
hp zbook	fonction = pro	laptop = hp zbook

3. Interface

Pour cette partie du projet, nous avons implémenté deux type d'interfaces :

- a. Une interface bureau à l'aide JavaFX.
- b. Une interface web à l'aide de Spring Boot et ReactJS.

Nous allons dans ce qui suit présenter les deux interfaces ainsi que leurs fonctionnements.

3.1. L'interface JavaFX

Au lancement du programme, il nous présente l'interface suivante (Voir Figure 2):

Les éléments principaux de l'interface sont les suivants :

1. Tableau récapitulatif de la base de connaissance.
2. L'ensemble des variables manipulées par le moteur d'inférence. On peut fixer la valeur de chaque variable individuellement avant l'application du chaînage avant, à l'aide des combobox dédiées.
3. Affichage des étapes du chaînage avant.
4. Bouton de lancement du chaînage avant.
5. Sélection de la base de connaissance.



Figure 2 : L'interface JavaFx

À l'application du chaînage avant, les étapes suivies sont affichées dans la zone 3, et les valeurs finales des variables sont affichées dans la zone 2 des variables (Voir Figure 3).



Figure 3 : L'interface après chaînage avant

Le bouton “ Bases ” permet de sélectionner une base de connaissance (voir Figure 4) :

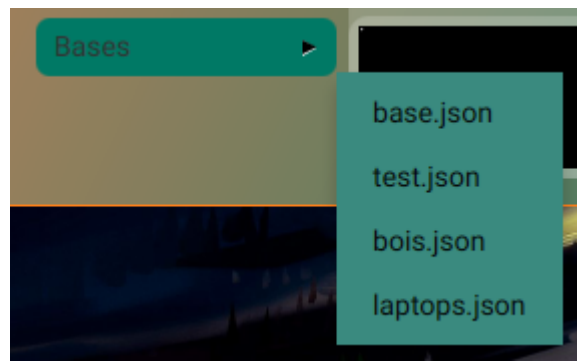


Figure 4 : Menu de sélection de base de connaissance

Exemple d'exécution de la base de Laptop (voir figure 5) :

Label	Antecedent clause	Consequent clause
acer nitro 5	fonction = jeux AND prix = pas cher	laptop = acer nitro 5
ultrabook	fonction = office AND autonomie = haute	office categorie = ultrabook
lenovo legi...	fonction = jeux	laptop = lenovo legion 5
asus chro...	office categorie = chromebook AND prix = pas cher	laptop = asus chromebook
hp probook	office categorie = ultrabook AND prix = pas cher	laptop = hp probook
chromebook	fonction = office	office categorie = chromebook
razer blade	fonction = jeux AND poids = léger	laptop = razer blade
apple mac...	fonction = pro AND utilisation pro = multimedia	laptop = apple macbook
lenovo thin...	fonction = pro AND utilisation pro = workstation	laptop = lenovo thinkpad
samsung c...	office categorie = chromebook	laptop = samsung chromebook
dell xps 13	office categorie = ultrabook	laptop = dell xps 13
hp zbook	fonction = pro	laptop = hp zbook

Filters on the right:

- prix: pas cher
- poids: [dropdown]
- fonction: office
- utilisation pro: [dropdown]
- office categorie: ultrabook
- laptop: hp probook
- autonomie: haute

Console logs:

```
[ Inference Engine ] : Conflict set : ultrabook, chromebook
[ Inference Engine ] : Selected rule : ultrabook
[ Inference Engine ] : Conflict set : hp probook, chromebook, dell xps 13
[ Inference Engine ] : Selected rule : hp probook
[ Inference Engine ] : Target variable found.
```

Figure 5 : Exemple d'exécution pour la base proposée


3.2. L'interface Web

Les applications web sont devenues le standard de développement des applications informatiques. Et il est nécessaire de savoir exploiter cette plateforme de déploiement d'applications.

C'est pour cette raison que nous proposons en plus de l'interface précédente, une interface Web développée à l'aide de la framework Java Spring Boot, et la framework React JS pour le côté front end (voir Figure 6).

Cette interface est composée des mêmes éléments que l'interface de Bureau précédente, mais avec un *touch and feel* plus moderne.

Le serveur Spring Boot écoute par défaut sur le port 8000, et l'application peut être chargée depuis l'URL *localhost:8000*.



Expert System

Current knowledge base - *laptops.json* -

Knowledge Base

Label	Antecedents clauses	Consequent clause
chromebook	fonction = office	office categorie = chromebook
ultrabook	fonction = office AND autonomie = haute	office categorie = ultrabook
asus chromebook	office categorie = chromebook AND prix = pas cher	laptop = asus chromebook
samsung chromebook	office categorie = chromebook	laptop = samsung chromebook
hp probook	office categorie = ultrabook AND prix = pas cher	laptop = hp probook
dell xps 13	office categorie = ultrabook	laptop = dell xps 13
razer blade	fonction = jeux AND poids = léger	laptop = razer blade
acer nitro 5	fonction = jeux AND prix = pas cher	laptop = acer nitro 5
lenovo legion 5	fonction = jeux	laptop = lenovo legion 5
apple macbook	fonction = pro AND utilisation pro = multimedia	laptop = apple macbook
lenovo thinkpad	fonction = pro AND utilisation pro = workstation	laptop = lenovo thinkpad
hp zbook	fonction = pro	laptop = hp zbook

Variables

fonction
office

office categorie
ultrabook

prix
pas cher

poids

autonomie
haute

utilisation pro

laptop
hp probook

Results

Step	Conflict Set	Selected Rule
1	ultrabook, chromebook	ultrabook
2	hp probook, chromebook, dell xps 13	hp probook
3	Target variable found	

Forward Chaining

Select a knowledge base
laptops.json

Ayoub Hammal | Mohamed Ait Amara

Figure 6 : Interface Web

Partie 2 : Système Multi-Agents

1. Conception

Nous exploitons dans cette partie une architecture multi-agents pour la gestion d'un magasin en ligne. Les requêtes d'achat sont d'abord envoyées vers un agent d'orchestration appelé agent *Indexer*, ce dernier consulte sa liste d'agents vendeurs ou *Sellers*, et communique avec chacun d'entre eux afin de bien servir ces requêtes. Donc, nous disposons de deux types d'agents : agent *Indexer* et agent *Seller*, et d'un certain nombre de types de communications et interactions entre ces agents.

L'agent *Indexer* reçoit la liste des agents *Sellers* en paramètre lors de son lancement. Quant aux agents *Sellers*, chacun d'eux reçoit le nom du fichier JSON de son magasin, qui contient les différents produits dont il dispose, ainsi que leurs caractéristiques, on y trouve aussi les promotions que peut proposer le magasin selon les périodes de l'année, et les pack ou "bundle"

de réduction qui sont proposés à l'achat de certain produits du magasin. Au lancement, chaque Seller charge les informations contenues dans ce fichier JSON dans sa mémoire de travail.

2. Les interactions

Il existe deux types d'événements auxquels l'agent Indexer doit répondre : Les recherches de produits, et les demandes d'achat.

À la réception d'une requête de recherche de produit, l'agent Indexer ajoute un comportement à exécution unique pour envoyer les filtres indiqués dans la requête aux agents Sellers dans un message ACL de CFP (CALL FOR PROPOSAL). Ensuite, il se met à attendre les réponses des Sellers (message avec performative PROPOSE), qu'il va regrouper et formater sous format JSON pour les envoyer au serveur.

En suivant le même schéma d'action, il traite aussi les requêtes de demandes d'achat. Cependant cette fois, en envoyant un message ACCEPT_PROPOSAL à seulement le Seller concerné par l'opération d'achat. Et puis, il attend la confirmation du Seller (performative INFORM), qui indique soit le bon déroulement de l'opération, soit un code d'erreur.

Quant au Seller, il dispose de deux comportements cycliques, le premier répond aux messages CFP en calculant la liste des produits compatibles avec les filtres de la requête à l'aide d'un moteur d'inférence. Et puis en les envoyant vers l'Indexer dans un message PROPOSE. Le deuxième comportement répond aux messages ACCEPT_PROPOSAL, le seller vérifie si bien tous les produits indiqués dans la requête d'achat sont disponibles et répond en conséquence soit avec le prix total après l'application des réductions s'il y en a, soit avec un message d'erreur dans le cas échéant. Le diagramme de séquence suivant résume ces interactions (voir Figure 7).

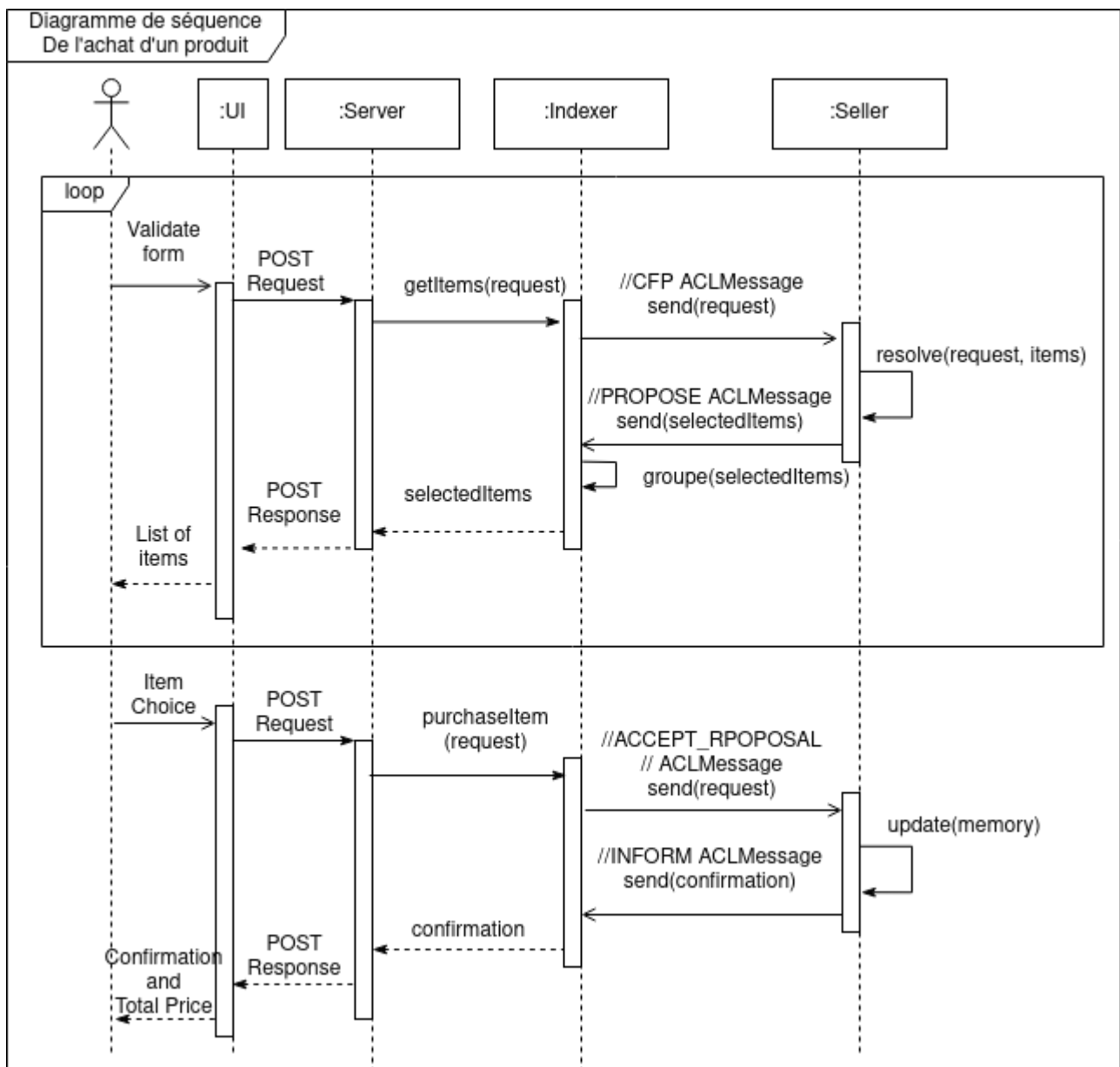


Figure 7 : Diagramme de séquence de l'achat d'un produit

3. Interface

Nous avons adopté une solution Web pour cette partie, en utilisant un serveur Java Spring Boot et ReactJS pour le développement de l'interface front end.

Au lancement de la page web, nous recevons l'interface d'accueil ci-dessous (voir Figure 8).

Sur cette interface, nous pouvons choisir la date du jour pour pouvoir bénéficier des promotions saisonnières, ainsi que la catégorie du produit qu'on cherche à acheter.

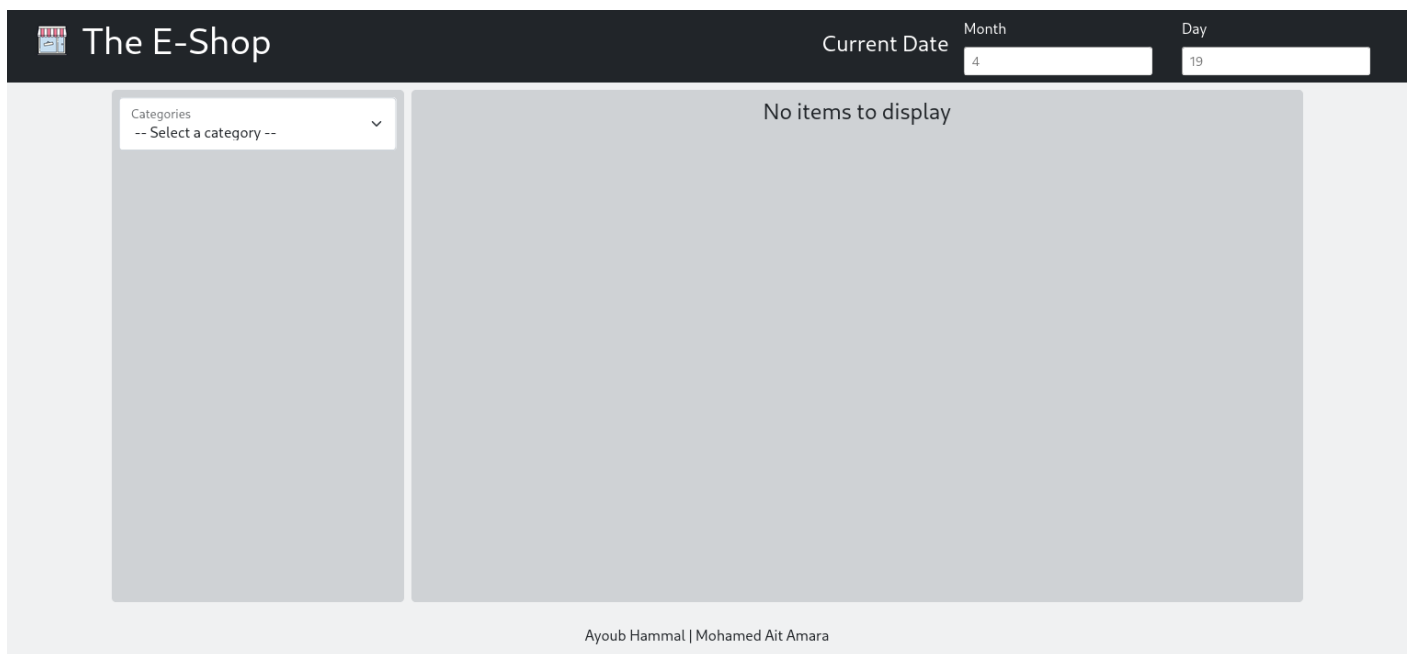


Figure 8 : Interface d'accueil

Nous allons dans ce qui suit suivre les étapes d'achat d'un laptop depuis l'application.

La figure 9 montre les résultats de la recherche d'un produit pour chacun des 3 magasins. On peut voir que le premier magasin propose une promotion pour le 22 Octobre, ainsi qu'un bundle facultatif à l'achat du laptop; l'achat supplémentaire d'une barrette de RAM pour une réduction de 10%.

Si la date n'est pas renseignée, la date du jour est utilisée comme date par défaut.

La quantité demandée par défaut est 1, mais celle-ci peut être changée dans la barre latérale de recherche.

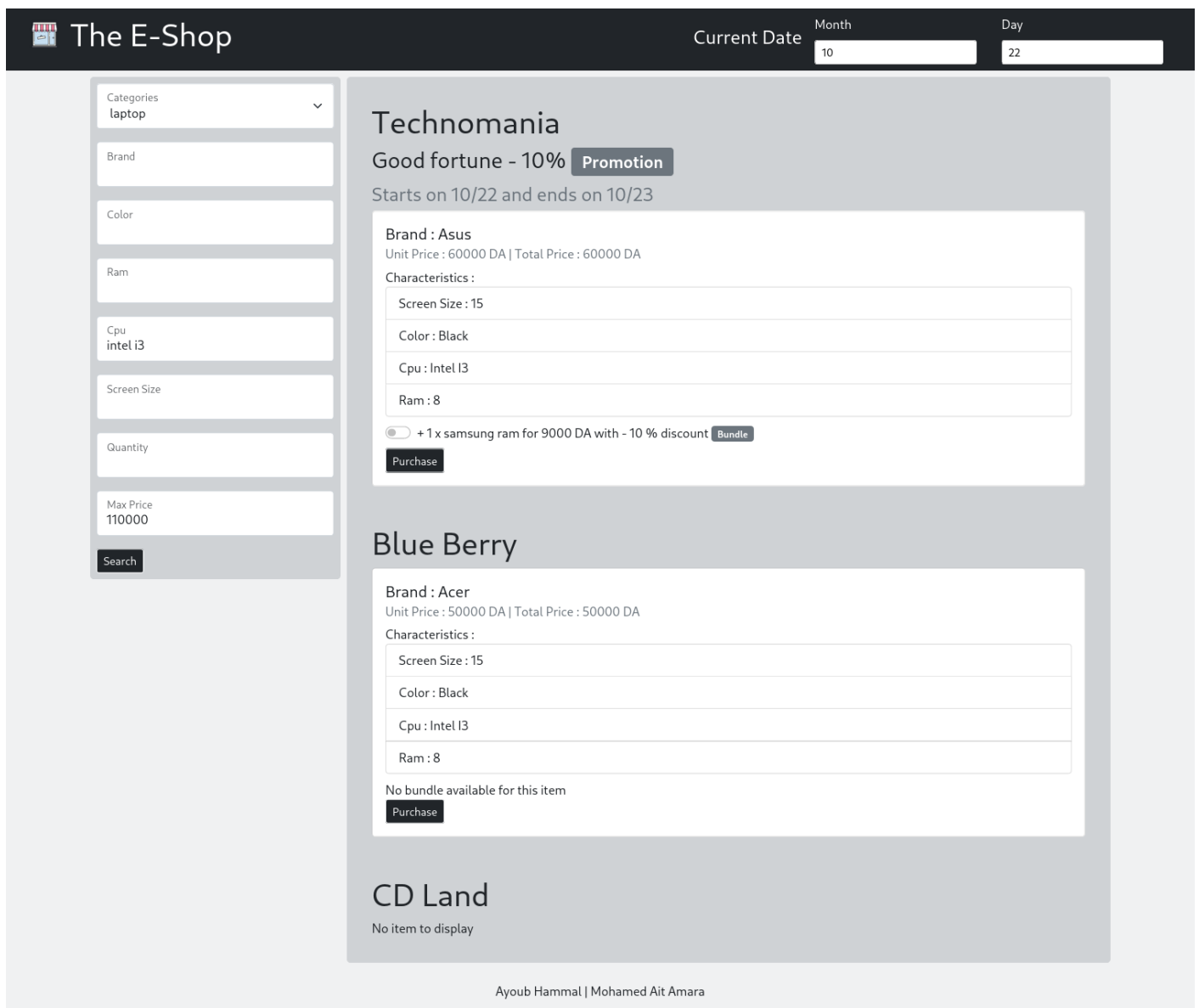


Figure 9 : La liste des produits après une recherche

Si on décide d'acheter le premier produit avec son bundle, l'application nous affiche le message de confirmation avec le coût total de l'achat après l'application de la promotion saisonnière et la promotion du bundle (voir Figure 10).

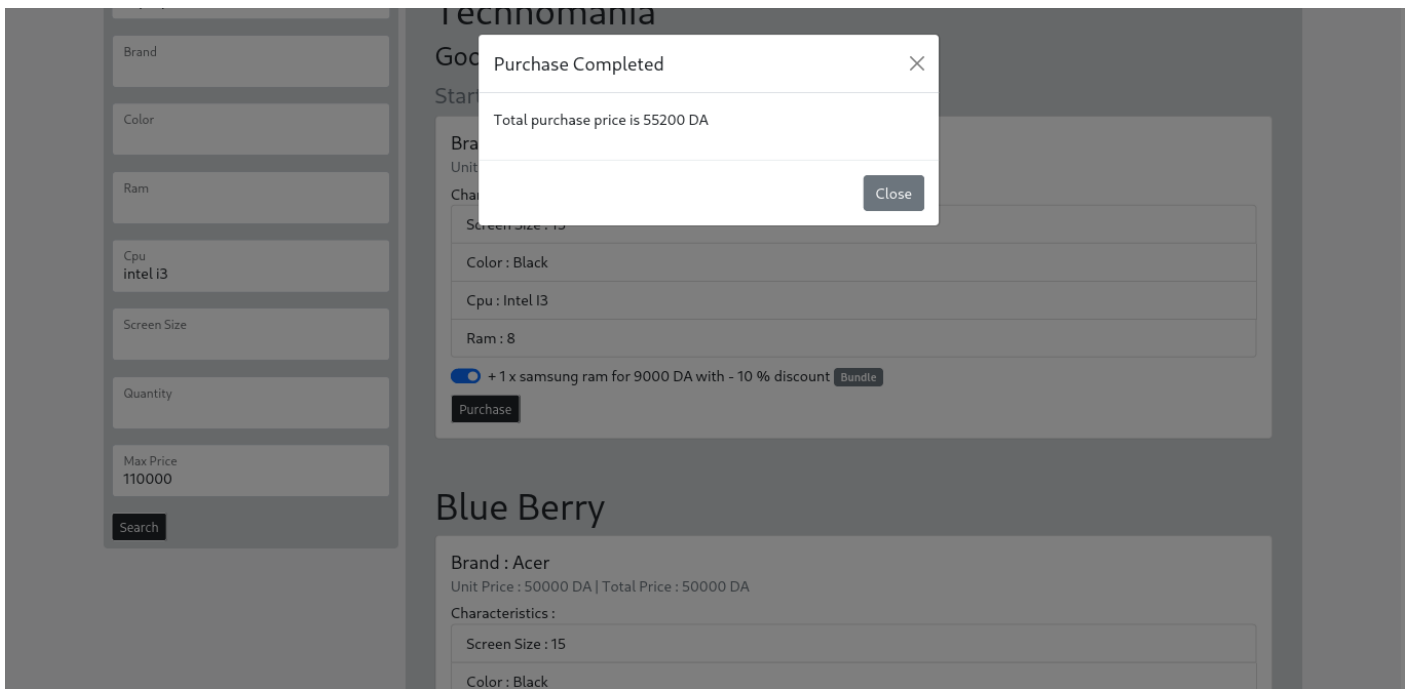


Figure 10 : Message d'achat

On peut ensuite continuer à utiliser l'application pour effectuer d'autres achats.

Du côté serveur, on peut retrouver les messages échangés entre les agents comme message log (voir Figure 11). Nous avons décidé de ne pas les afficher sur l'interface graphique car ceci serait encombrant pour l'affichage.

Les échanges entre les agents sont effectués à l'aide du format JSON qui encode efficacement les informations à transmettre, et permet d'avoir une structure consensuelle pour la représentation des données.

```

Opération de recherche de produit
[ /api/items | request ] : {"category":"laptop","filters":[{"feature":"cpu","condition":"=","value":"intel i3"}, {"feature":"price","condition":"<","value":"110000"}],"date":{"month":"10","day":"22"}}
[ /api/items | response ] : [{"seller":"seller3","promotions":[],"name":"CD Land","items":[]}, {"seller":"seller1","promotions":[{"start day":"22","end month":"10","discount":"10","id":"1","start month":"10","title":"Good fortune","end day":"23"}], "name":"Technomania","items":[{"screen size":"15","quantity":"5","color":"black","total price":"60000","price":"60000","cpu":"intel i3","id":"1","brand":"asus","bundle":{"item":{"quantity":"4","price":"9000","id":"1","brand":"samsung","capacity":"8"},"quantity":"1","discount":"10","id":"1","category":"ram"},"ram":"8"}]}, {"seller":"seller2","promotions":[],"name":"Blue Berry","items":[{"screen size":"15","quantity":"3","color":"black","total price":"50000","price":"50000","cpu":"intel i3","id":"1","brand":"acer","ram":"8"}]}]

Opération d'achat du produit
[ /api/item/purchase | request ] : {"seller":"seller1","details":{"category":"laptop","id":"1","quantity":"1","bundle":"true","date":{"month":"10","day":"22"}}}
[ /api/item/purchase | response ] : {"total price":"55200","status":"completed"}
  
```

Figure 11 : Communication entre agents

Conclusion

Nous avons dans ce travail réalisé deux applications pour répondre à des besoins commerciaux. La première application est une plateforme d'exploitation de systèmes experts employant la technique de chaînage avant. La deuxième application est une plateforme multi-agents de commerce électronique où la gestion des magasins est déléguée à chaque agent.