

Ministère de l'enseignement supérieur et de la recherche scientifique
Université Saad Dahlab Blida 1



FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Rapport de projet tutoré

L3 SIQ

Thème

***Réalisation d'une application broker pour la sélection de
services Clouds de type SaaS***

Domaine : MI
Filière : Informatique
Spécialité : SIQ

Encadré par :

M^{me} MANCER Yasmine

Réalisé par :

AFIR Sofiane

BOUMECHTA Mohamed Issam-Eddine

Année Universitaire : 2018/2019

PLAN

1. Introduction au service cloud
2. Problématique
3. Analyse des besoins
4. Revue de littérature
5. Conception
 - 5.1 Acteurs
 - 5.2 Fonctions de chaque acteur
 - 5.3 Détails de chaque action
6. Réalisation
7. Conclusion
8. Référence bibliographique

1.INTRODUCTION AU SERVICE CLOUD

Le Service cloud consiste à exploiter la puissance de calcul ou de stockage de serveurs informatiques distants par l'intermédiaire d'un réseau, généralement Internet. Les serveurs sont loués à la demande selon des critères techniques (puissance, bande passante, etc.), mais, également, au forfait .

Les principaux services proposés en *cloud computing* sont :

- **le IaaS (Infrastructure as a Service) :** C'est le service de plus bas niveau. Il consiste à offrir un accès à un parc informatique virtualisé. Des machines virtuelles sur lesquelles le consommateur peut installer un système d'exploitation et des applications .
- **le PaaS (Platform as a Service) :** Dans ce type de service, situé juste au-dessus du précédent, le système d'exploitation et les outils d'infrastructure sont sous la responsabilité du fournisseur. Le consommateur a le contrôle des applications et peut ajouter ses propres outils.
- **le SaaS (Software as a Service) :** Dans ce type de service, des applications sont mises à la disposition des consommateurs. Les applications peuvent être manipulées à l'aide d'un navigateur Web ou installées de façon locative sur un PC, et le consommateur n'a pas à se soucier d'effectuer des mises à jour, d'ajouter des patches de sécurité et d'assurer la disponibilité du service. Un fournisseur de *software as a service* peut exploiter des services de type *platform as a service*, qui peut lui-même se servir de *infrastructure as a service*.

2.PROBLÉMATIQUE:

Notre travail consiste en la création d'une application Java qui permet de choisir automatiquement un service cloud de type SAAS le plus adapté au besoin de l'utilisateur qui spécifie ses contraintes non fonctionnel suivant des algorithmes.

3. ANALYSE DES BESOINS

Sélection d'un service cloud de type SAAS.

4. REVUE DE LITTÉRATURE

Ce domaine n'a pas reçu beaucoup d'attention de la part des chercheurs et peu de littérature scientifique a été publiée, cela est dû au fait qu'il est à ces débuts. Parmi les majeurs obstacles de sélection du service cloud est la multiplicité et la diversité des offres cloud ce qui rends la comparaison des services entre eux très difficile.

5. CONCEPTION

ACTEURS

Broker.
Fournisseur.
Client.

FONCTIONS DE CHAQUE ACTEUR

Broker : il gère les comptes de chaque fournisseur et de chaque client et valide une commande d'un client.

Fournisseur : Create, Update et Delete d'un service.

Client : Consulter les services proposé par un fournisseur et voir le résultat de filtre de sélection de ce dernier.

DÉTAILS DE CHAQUE ACTION

Broker:

- **Consult** : Affiche les informations des trois tables de la base de données(Account,Fournisseur,Client).
- **Update**: Modification des informations du fournisseur ou bien du client.
- **Add Account**: Confirmation des demandes d'inscription d'un fournisseur ou bien d'un client et l'ajout à une des différents tables de la base de données, petite remarque si la demande est d'un fournisseur alors c'est au broker d'écrire le domain manuellement .
- **Delete**: Suppression d'un compte existant d'un fournisseur ou bien client.
- **Validate**: Valide les commandes des client, pour chaque validation un contrat est écrit contenant les informations du fournisseur et du client, après le nombre de vente de fournisseur concerné dans le contrat est augmenté et en même temps il met à jour la contrainte « confiance » automatiquement suivant cette formule :
 nbV=nombre de vente ;
 diff=la différence entre la date actuelle et la date de service ;
 conf=la contraint confiance ;
 si(nbV<5 && Diff>20 && conf>0)alors
 conf=conf-1;
 sinon
 si(nbV == 5 && conf<10)alors
 conf+=1;}
 sinon
 si(nbV>6 && Diff<20 && Diff>10 && conf>0)alors
 conf=conf+1;
 sinon
 si(nbV == 6 && conf<10)alors
 conf=conf+1;
 si(nbV>20)alors
 conf=(conf+1)*3/2;
 si(conf>10)alors
 conf=10;.

Remarque : le nom d'utilisateur et le mot de passe est admin. S'il y a une erreur d'introduction de nom d'utilisateur ou de mot de passe, une nouvelle fenêtre avec le message suivant :
(« incorrect username or Password ») apparaît.

Fournisseur :

- **Create/Update :** Permet de créer un service en entrant ses spécifications (réputation, expérience, disponibilité) qui sont des contraintes non fonctionnel positive, (coût, temps de réponse, temps moyen de représentation et risques) qui sont des contraintes non fonctionnel négative, ses spécifications vont être placées directement dans le fichier WSDL dans la balise <types><propriétés> qui seront transmises dans la base de données correspondante, une petite précision la contrainte non fonctionnelle « confiance » est gérée automatiquement en fonction du nombre de services vendus.
- **Delete :** Permet de supprimer un service cloud définitivement de la base de données.

Remarque : le nom d'utilisateur et le mot de passe est inclus dans la table de base de données fournisseur.

Client:

- **Consult :** Permet de consulter les informations des services qui proposent un fournisseur, il a à sa disposition tous les noms des fournisseurs disponibles avec leurs domaines.
- **Filtre 1:** Comparaison entre différents services cloud dans un certain domaine, le résultat est donné suivant la valeur maximum des contraintes non fonctionnelles positives et la valeur minimum des contraintes non fonctionnelles négatives, pour calculer le max on utilise cet algorithme :
int i = « la valeur à calculer son max choisit par le client » ;
max=0 ;
si(i>max) alors max=i ;
et on le fait autant qu'il y aura un service ;
pour calculer le min on utilise cet algorithme :
int i = « la valeur à calculer son min choisit par le client » ;
min=0 ;
si(i<min) alors min=i ;
et on le fait autant qu'il y aura un service ;
- **Filtre 2 :** Une moyenne est calculée suivant le choix du client et la comparaison se fait par rapport aux différents services cloud et le meilleur résultat sera validé, on utilise cet algorithme :
max[] = « toutes les contraintes non fonctionnelles positives » ;
min[] = « toutes les contraintes non fonctionnelles négatives » ;
moy = max[] + (10 - min[]) ;
moy_max=0 ;
si(moy>moy_max) alors moy_max=moy ;

- **MCDM** : Un Algorithme provenant d'un article scientifique est utilisé. Son principe est le suivant :
 - récupère toutes les fonctions des services provenant des différents fournisseurs sous forme de chiffre entre 1 et 10 puis les mets dans une matrice nommée A.
 - récupère les différents critères qu'un client veut voir dans les services proposés dans un tableau nommé R.
 - récupère les poids de chaque critère car le client veut des fois donner plus de priorité à un critère plutôt qu'à un autre. On met les valeurs dans un tableau nommé W.
 - la formule est la suivante : $\text{exponentielle}(-(A[i]-R[i])*W[i]))$ ou i représente l'indice de case actuelle.
 - chaque résultat de la formule on le met dans un tableau nommé F.
 - on prend les valeurs du tableau F et on cherche la valeur la plus minimale et on rend au client le nom du service ayant le meilleur résultat et on affiche ses critères.

- **AHP** : Un Algorithme provenant d'un article scientifique est utilisé. Son principe est le suivant :
 Petit remarque : on utilise juste 3 contraintes pour moins de complication et pas plus de 3 cloud service.

- le client spécifie la matrice suivante, la ligne diagonale sera rempli de 1, exemple : la disponibilité par rapport au réputation est 8 et la réputation par rapport au disponibilité est 1/8. On remplit les restes des champs en suivant cette méthode

reputation	disponibilité	experience
1	0.125	0.333
8	1	3
3	0.333	1

- calcule du poids

$$(1.000 \times 0.125 \times 0.333) = (0.042)^{(1/3)} = 0.347;$$

$$(8.000 \times 1.000 \times 3.000) = (24.000)^{(1/3)} = 2.884;$$

$$(3.000 \times 0.333 \times 1.000) = (1.000)^{(1/3)} = 1.000.$$

(puissance 1/3 car on a 3 constraint)

3rd root of product
0.347
2.884
1.000
4.231

- calcule de priorité

$$(0.347 / 4.231) = 0.082;$$

$$(2.884 / 4.231) = 0.682;$$

$$(1.000 / 4.231) = 0.236.$$

Priority vector
0.082
0.682
0.236
1.000

- priorité*sum

	reputation	disponibilité	expérience
	1.000	0.125	0.333
	8.000	1.000	3.000
	3.000	0.333	1.000
Sum	12.000	1.458	4.333
Sum*PV	0.983	0.994	1.024

Priority vector
0.082
0.682
0.236
1.000

$$12.000 \times 0.082 = 0.983;$$

$$1.458 \times 0.682 = 0.994;$$

$$4.333 \times 0.236 = 1.024.$$

$$\text{Lambda-max} = 0.983 + 0.994 + 1.024$$

n=nombre de contraint

- calcul de consistency index

$$\text{CI} = (\text{Lambda-max} - n) / (n - 1)$$

$$\text{CI} = (3.002 - 3) / (3 - 1) = (0.002) / (2) = 0.001$$

Consistency Ratio (CR) = Consistency Index (CI) / Random Index (RI)

$$\text{CR} = \text{CI} / \text{RI} = 0.001 / 0.58 = 0.001$$

n	Random Index (RI)
1	0.00
2	0.00
3	0.58
4	0.90
5	1.12
6	1.24
7	1.32
8	1.41
9	1.45

2ème étape

- on récupérer les services cloud et on les met dans une matrice suivant cette algorithme :

si(i=j) alors

T[i][j]=1

sinon

si (i<j) alors

valeur=cs[i]/cs[j]

si(valeur>0)alors

fonction() // cette fonction détermine est ce que la valeur est avec une virgule si oui alors on cherche la valeur rapprochée

T[i][j]=valeur

finsi

sinon

valeur=cs[i]/cs[j]

si(valeur>1)

```

fonction(valeur) // cette fonction détermine est ce que la valeur est avec
                  une virgule si oui alors on cherche la valeur approché
sinon
fonction2(valeur)//cette fonction cherche la valeur la plus proche au
              (0.5,0.333,0.25,0.2,0.166,0.142,0.125,0.111)
fini
T[i][j]=valeur
fini

```

- on refait les mêmes étapes déjà faites sur les 3 constraints pour chaque service cloud
- on multiplie les priorités de chaque service avec la priorité de matrice du client, chaque service aura un score. Le meilleur score sera choisit et retransmit au client

Remarque1 : le nom d'utilisateur et le mot de passe est inclut dans la table de base de données client.

Remarque2 : le bouton « valider » situé dans toute les actions du client qui fait la confirmation d'achat d'un service cloud puis attende juste la confirmation de broker qui lui attribut un contrat.

Base de données :

Composée de quatre tables :

1ère table : « fournisseur » contenant 7 champs :

- fournisseur_clé : représente un ID de chaque fournisseur .
- fournisseur_nom : le nom du fournisseur.
- fournisseur_password : le mot de pass du fournisseur.
- fichier : fichier d'une extension WSDL.
- domain_nom : le domaine choisi par le fournisseur.
- fournisseur_date : date de création du service cloud.

2ème table : « Client » contenant 3 champs

- client_nom : le nom du client.
- client_password : le mot de pass d'un client.
- client_points : les points gagné après chaque achat
comme le fournisseur il a droit qu'à un seul compte.

3ème table : « Account » contenant 5 champs

- account_id : id de chaque inscription.
- account_nom : nom de la personne inscrit.
- account_password : mot de pass de la personne inscrit.
- account_detail : détail de la personne inscrite, soit fournisseur soit client.
- domain_nom_account: si le account_detail est fournisseur alors le domain sera rempli sinon il sera vide car le client n'a pas de domain.

Petite précision si un fournisseur ou bien un client n'a pas de compte, il doit donc s'inscrire. Les demandes d'inscriptions sont envoyés à la table « Account » et c'est au broker de les valider ou non. S'il valide, les demandes seront automatiquement envoyés aux tables « fournisseur » ou « client », suivant les détails écrit dans la demande.

4ème table : « validation » contenant 5 champs

- id : id de chaque validation.
- client_nom : le nom du client .
- client_password : le mot de pass d'un client.
- fournisseur_nom : le nom du fournisseur.
- fournisseur_domain : le domain du fournisseur.

Remarque :chaque fournisseur a droit d'avoir un seul compte dans chaque domaine.

6. RÉALISATION

On a travaillé sur Java EE et java se.

On a utilisé XAMPP pour la base de donnés.

On a utilisé TOMCAT pour le serveur.

On a utilisé la méthode MCDM et plus précisément le formalisme décrit dans [1].

On a utilisé la méthode AHP et plus précisément le formalisme décrit dans [2].

7. CONCLUSION

Dans ce domaine, il est difficile de concevoir des algorithmes capables de faire une sélection optimale d'un service cloud. Cette difficulté est due à différents paramètres parmi lesquels la multiplicité et la diversité des services cloud offerts.

8. REFERENCE BIBLIOGRAPHIQUE

- [1] Z. ur Rehman, F. K. Hussain, and O. K. Hussain, "Towards Multi-criteria Cloud Service Selection," in 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2011, pp. 44–48.
- [2] Mingrui Sun, Tianyi Zang, Xiaofei Xu, and Rongjie Wang, "Consumer-Centered Cloud Services Selection Using AHP," in 2013 International Conference on Service Sciences (ICSS), 2013, pp. 1–6.

