DING Feng
JIN Hong
WANG Yuqi

Rapport final serveur d'entreprises

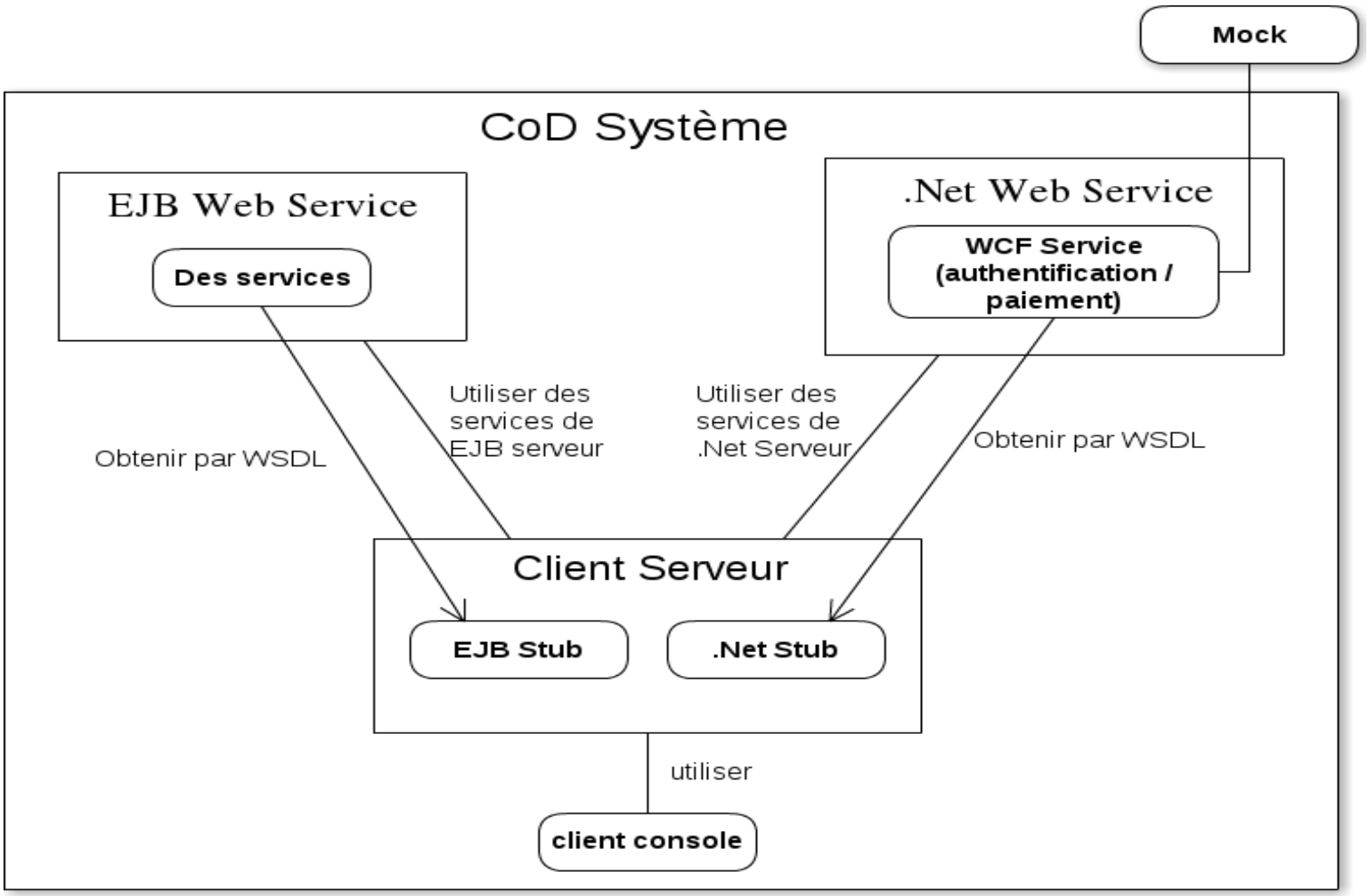
Introduction

Dans cette vertion, nous vous présenterons notre architecture, les choix de conception fait au niveau global (Java et .Net) et ce qu'on à réalisé pour chaque scénario.

Architecture générale

Notre application repose sur deux serveurs d'applications, un pour l'écosystème Java et un second pour l'écosystème .Net. La partie Java Beans peut fournir des services de CookieFactory et la partie .net permet de s'authentifier et faire une paiement. Le EJB serveur prend des interfaces de .Net serveur par WSDL.

Les utilisateurs peuvent communiquer avec notre système par le client console. Ils peuvent première accéder à notre système par EJB Serveur. Et ensuite, Ils peuvent s'authentifier ou entrer directement dans le CoD système. Puis, Il peut utilise des services fourni par le serveur pour différente type de utilisateur (par exemple créer un boutique par la responsable de la marque).



Choix d'architectures

- **Client**

Quand l'utilisateur entre notre système, une fois qu'il insère son mot de passe et son login, le partie client stocke les informations d'utilisateur jusqu'à l'utilisateur arrète sa session. Ce mot de passe et son login sont fournis par la partie .Net, et le client partie guide l'utilisateur directement dans les fonctionnalité que l'utilisateur peut faire grâce aux informations d'utilisateur.

- **Serveur Java**

Dans notre serveur Java, il y a trois grandes choses. Des web services, un bean initial et des JSFs.

Pour la partie de JSF, il n'y a pas d'authentification puisqu' il ne faut pas créer la couche présentation complète. On a créé deux pages WEB étant la page de la création de boutique et la page de la liste des commandes et de chiffre d'affaire dans une boutique.

On a créé aussi un bean initial qui crée quelque boutiques, quelque façcons de cussine, quelque ingrédients, etc en avance, pour bien présenter des service de EJB.

Pour la partie de web service, on a découpé des service par les utilisateurs. Chaque bean de web service contient des fonctionnalités qu'un type d'utilisateur peut faire.

- **Serveur .NET**

Le Serveur .NET ayant 2 types principal de fonctionnalités est dans le class WcfServiceTCF.ServiceTCF. Une fonction permet le payment et l'historique de payment et la gestion des informations de payment et l'autre permet de gérer les comptes. Ces deux fonctionnalités permettent de réaliser toutes les fonctionnalités demandées pour The Cookie Factory.

Les services sont exposés par un WSDL du service WCF. Pour le client .Net, on a utilisé le WCF Securité, malheusement, il dépend le wshttpbinding, mais le client JAVA veut le basichttpsbinding, donc on a créé un autre service : ServiceTCF.ServiceTCF pour le client JAVA. Du coup, la communication entre les clients de la coté EJB et le serveur .NET est similaire à la communication entre ejb et le client JAVA.

- **Mock**

L'interaction avec le service bancaire tierce ne sera pas dans le périmètre du projet et sera remplacé par un mock. On a just simulé les étapes de payment. Si le client a un compte, il peut choisit les information de payment directement par la numéro de carte et le cryprogramme pour payer et il peux aussi utiliser une nouvelle carte et puis le système va sauvegarder cette information automatiquement. Si le client n'a pas de compte, il doit utiliser une nouvelle carte. On a définit le longeur de numéro de carte doit être 8 et le

longeur de cryprogramme doit être 3. Si tous sont bon, le payment est succès, sinon, il va envoyer les message d'erreur.

Pour chaque scénario comment notre architecture permet de le réaliser

- **Faith**

Faith entre son login et mot de passe, mettons "admin" "admin" . Une fois cette opération faite, il lui suffit de choisir l'option correspondante pour créer une boutique ou accéder aux statistiques de vente globale de sa marque. Il ne lui est pas possible de faire les opérations au-dessus sans connexion préalable

- **Carter**

Carter est un responsable du magasin. Il entre son login et mot de passe, mettons "Carter" "123". Quand il se connecte le système, le .Net service peut founir sa rôle et son magasin, donc il accéde à son magasin directement pour regarder la liste des commandes passées pour son magasin, ainsi que ses statistiques de ventes

- **Tom**

Tom est un cliente avec compte. Il entre son login et mot de passe, mettons "Tom" "123". Quand il se connecte le système, il peut faire une commande avec son magasin préféré et sa recette préférée ou faire une commande normale. Son id de compte est stocké dans la partie .Net et ses préférences sont stockées dans la partie de EJB. Il peut aussi regarder ses commandes historique qui sont stockées dans la partie .Net.

- **Brenda**

Brenda peut entrer directement dans notre Cod Système comme un cliente occasionnelle. Il peut faire une commande ou créer un compte en insérant son login, sa code passe, son magasin préféré et sa recette préférée. Une fois qu'il crée son compte, il devient un cliente régulier comme Tom.

Conclusion

Pour tous les sénarios, il faut faire une authentification en premier par la partie .Net. Ensuite, grâce à la rôle d'utilisateur, le client serveur lui donne des services correspondants de EJB.