Architectures distribuées
Travaux Pratiques (N°1)

1 Contexte

La société HighTunes décide de développer un nouveau système d'informations qui permettra aux particuliers de télécharger différents fichiers multimédia : musiques, clips, films, … Ce système devra permettre à différents types de clients de se connecter : **clients légers** (navigateurs, téléphones, baladeurs), **clients lourds** (application fournie par HighTunes).

Ce système sera également inter-connecté avec ceux de certains de ses fournisseurs auprès desquels HighTunes s'approvisionne.

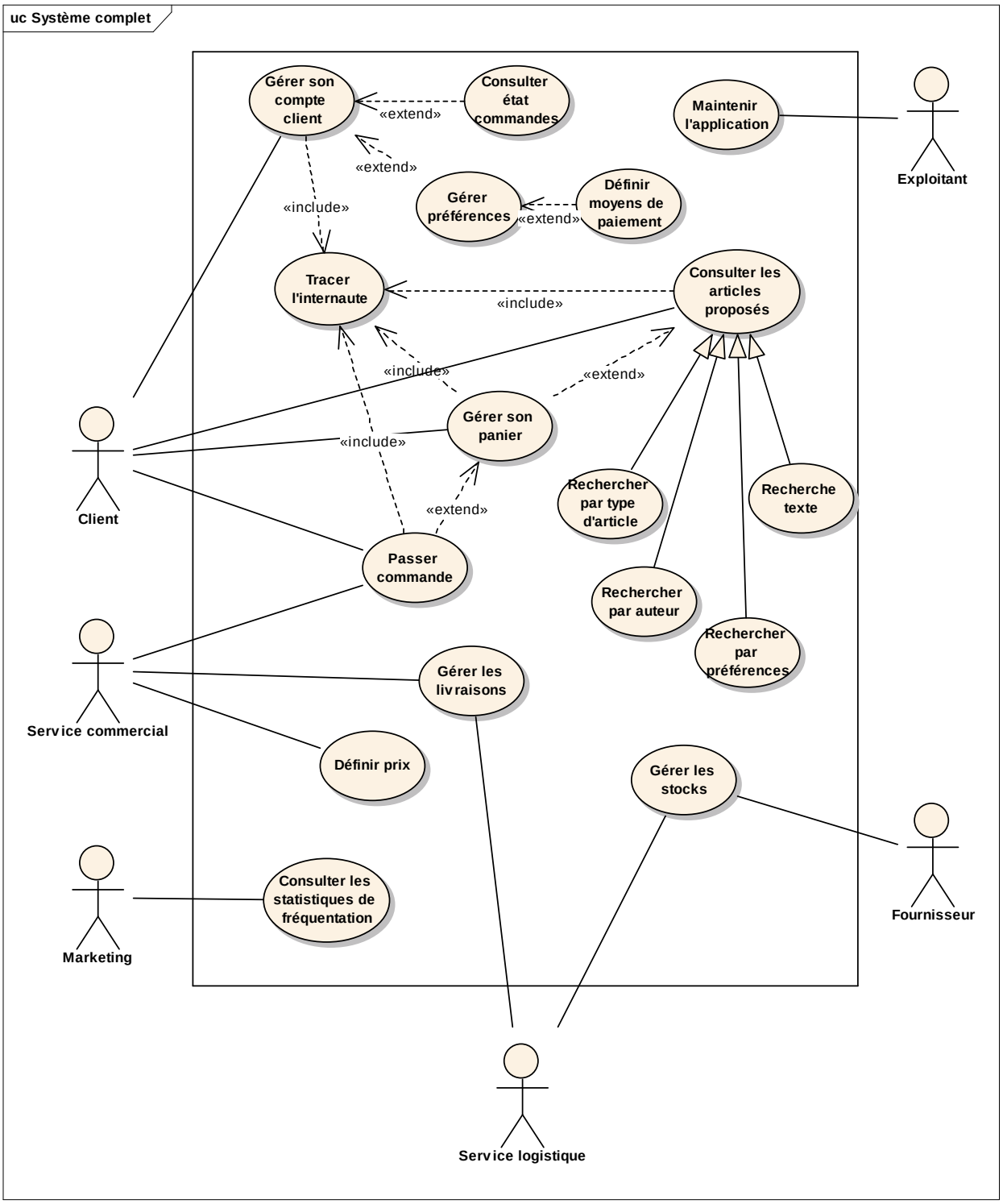
Pour limiter le risque lié à l'introduction de ce nouveau système, HighTunes souhaite conserver ses applications et matériels existants, qui sont validés et fonctionnent correctement depuis longtemps.

Le service Marketing de HighTunes met en place parallèlement à l'ouverture de son site de commerce en ligne une application de « tracking ». Cette application est chargée de tracer l'activité des internautes sur le site, et aussi d'enregistrer la provenance de ces internautes. En effet, des bannières publicitaires ont été créées et il est intéressant de connaître leur efficacité.

HighTunes est présente dans de nombreux pays. De ce fait, le nombre de clients potentiels est important.
En outre, le système doit être toujours disponible car il est utilisé depuis plusieurs fuseaux horaires différents.

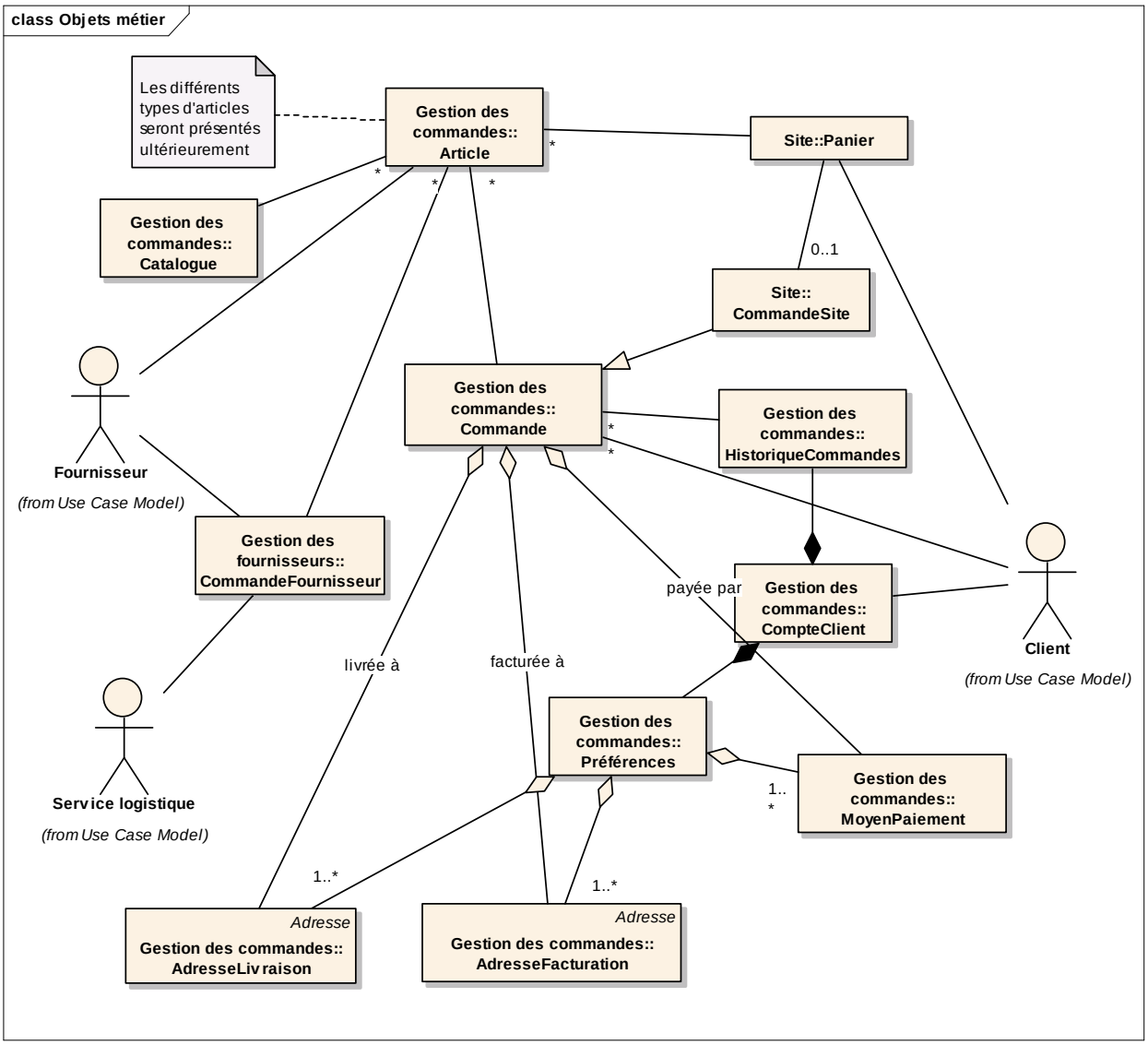
1.1 Analyse fonctionnelle

Une analyse fonctionnelle des besoins a été menée à partir du cahier des charges. Il en ressort que les cas d'utilisation du futur système peuvent être présentés de la façon suivante :



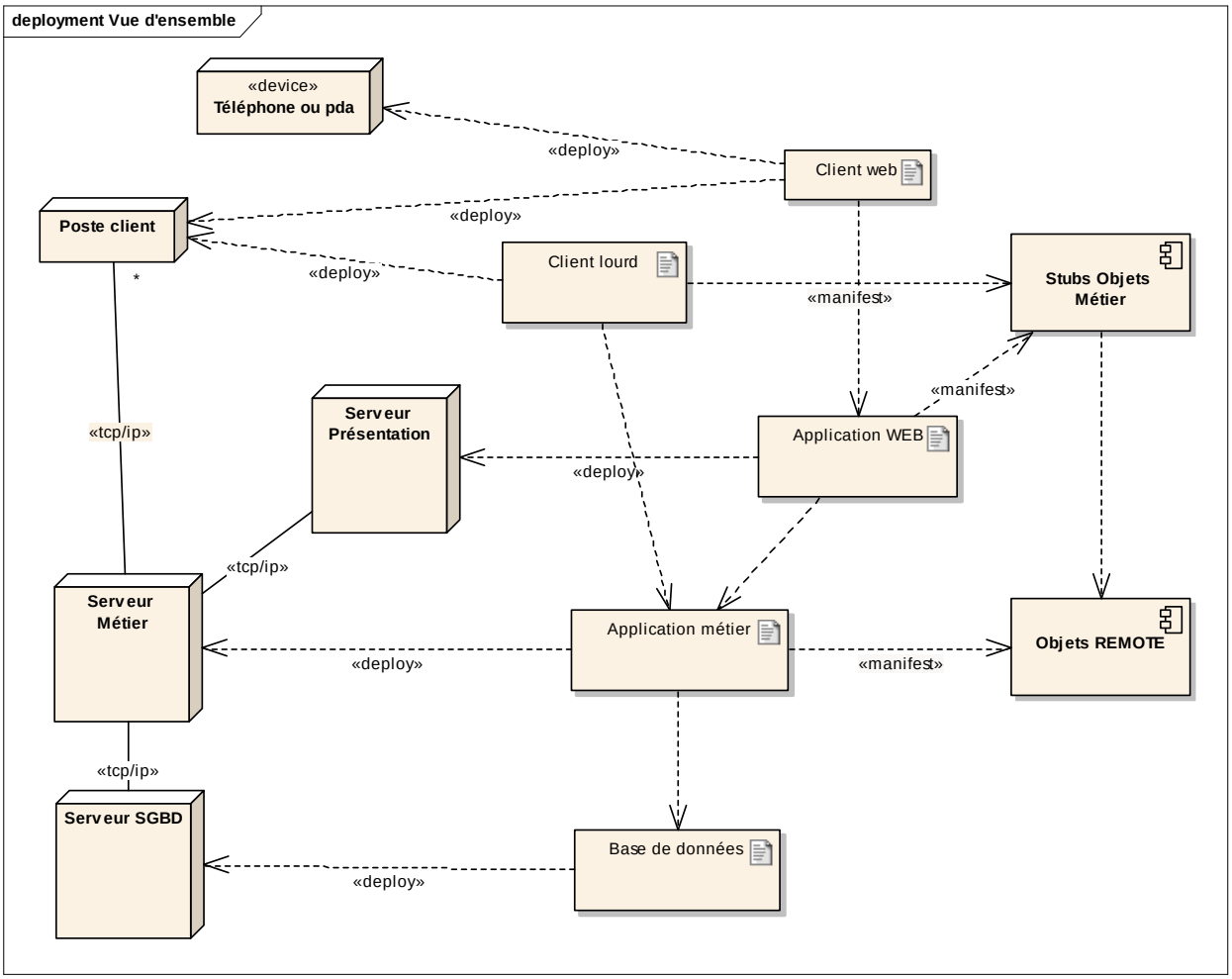
1.2 Objets Métier

Une analyse des objets métier fournit le diagramme de classes suivant :



1.3 Architecture générale

Le diagramme de déploiement suivant présente l'ensemble des choix en matière de déploiement :



Certaines associations ne faisant pas partie du périmètre de notre étude ne sont pas représentées.
Le périmètre est le suivant :

- Nœuds « Serveur Métier », « Poste Clients »
- Composants « Objets REMOTE », « Stubs Objets Métier »
- Artefacts « Application métier », « Client lourd »

1.4 Documentation utile

- Tutorial RMI : http://java.sun.com/docs/books/tutorial/rmi/index.html

2 Travaux pratiques RMI

Pour chaque énoncé présenté ci-après, vous devrez :

- concevoir une solution et la présenter à l'aide d'un diagramme de classes UML de conception détaillée. Les informations portées par ce diagramme devront être :
  - toutes les classes
  - toutes les interfaces
  - la nature des classes et interfaces (Remote, Serializable, …)
  - les associations et autres liens

- réaliser la solution
- réaliser une application de test montrant son bon fonctionnement

2.1 Catalogue

Les différentes applications clientes doivent être en mesure d'obtenir le catalogue des produits à vendre. Il vous est donc demandé de concevoir le mécanisme correspondant. Votre Catalogue doit offrir les services suivants :

- obtention du nombre total d'articles dans le système
- obtention de toutes les informations articles pertinentes :
    - clé
    - description textuelle
    - prix unitaire
    - date de disponibilité (si non valuée, l'article est disponible)
- obtention des informations d'un article à partir de sa clé. Si aucun article ne correspond à la clé passée en paramètre, l'opération doit renvoyer une exception à l'appelant.

2.1.1 Diagramme de classes UML

Fournir un diagramme de classes de conception détaillée, montrant notamment les interfaces et les classes.

2.1.2 Réalisation

L'application cliente est de type application Java autonome, sans téléchargement de classes RMI.

2.1.3 Questions

- Combien d'instances de Catalogue doit-on créer sur le serveur pour un client ?
- Combien d'instances de Catalogue doit-on créer sur le serveur pour tous les clients ?

2.2 Panier

Cette étape du développement consiste à concevoir et réaliser le mécanisme de gestion du Panier du client. Ce mécanisme doit permettre :

- l'instanciation d'un Panier pour chaque client
- la conversion d'un Panier en commande
- l'ajout d'articles au Panier
- la modification des quantités des articles stockés dans le Panier
- la suppression d'un article du Panier
- le calcul du montant global du Panier
- la destruction d'un Panier

2.2.1 Diagramme de classes UML

Fournir un diagramme de classes de conception détaillée, montrant notamment les interfaces et les classes.

2.2.2 Réalisation

L'application cliente est de type application Java autonome, sans téléchargement de classes RMI.

2.2.3 Questions

- Combien d'instances de Panier doit-on créer sur le serveur pour un client ?
- Combien d'instances de Panier doit-on créer sur le serveur pour tous les clients ?

2.3 RMI et sécurité

2.3.1 SecurityManager

Vous devez reprendre le code développé pour les deux TP précédents et activer le SecurityManager du serveur et celui du client.

Documentation : $JDK/docs/technotes/guides/security.index.html

2.3.2 Contrôle du port socket utilisé pour la communication RMI

Pour pouvoir paramétrer le firewall qui sera déployé côté serveur, un numéro de port spécifique est imposé pour la communication entre le client et le serveur. Il vous est demandé de modifier le code ainsi que les fichiers .policy en conséquence.

2.4 Téléchargement des classes RMI

Dans ce TP, il vous est demandé de reprendre le code et les fichiers de paramétrage développés jusqu'à présent et de les adapter afin que le client puisse s'exécuter depuis une installation différente de celle du serveur :

- Même machine, mais CLASSPATH différent
- Autre machine

2.5 Pattern ServiceLocator

Concevoir et réaliser un mécanisme permettant d'encapsuler l'accès au service de nommage RMI.

2.6 Bootstrap RMI

Il vous est demandé d'écrire un client autonome générique capable de :

- Télécharger l'application à exécuter sur le poste client depuis une autre machine
- Lancer son exécution

Ce « bootstrap » doit pouvoir s'exécuter uniquement à partir des classes de base du JDK et ne nécessiter aucune autre installation locale.

2.7 Pool d'objets serveur

L'objectif de ce TP est de concevoir et réaliser un système de gestion d'un pool d'objets RMI qui puisse être mis-en-œuvre pour le Panier, mais également pour n'importe quel autre objet RMI serveur.
Ce pool doit proposer les services suivants :

- Création avec un nombre d'objets prédéfini qui seront instanciés au lancement du processus serveur
- Obtention d'une référence distante sur l'un des objets du pool
- Restitution de la référence au pool

2.8 Activation des objets serveur

L'objectif de ce TP est de transformer l'objet Catalogue en un objet 'activable' RMI. Il vous est donc demandé de réaliser toutes les transformations nécessaires.

2.9 Callbacks

Vous devez ajouter au système Hightunes un serveur de messages qui permet aux clients qui s'abonnent de recevoir des informations en provenance du serveur.
Ce serveur propose les opérations suivantes :

- Abonnement d'un client
- Désabonnement

L'envoi des messages à tous les abonnés est réalisé en « push » par le serveur.