IPST-CNAM
Intranet et Designs patterns
NSY 205
Jeudi 5 Février 2015

Durée : **2 h 30**
Enseignant : LAFORGUE Jacques

1ère Session NSY 205

1ère PARTIE – SANS DOCUMENT (durée: 1h15)
CORRECTION

1. QCM (35 points)

Mode d'emploi :

Ce sujet est un QCM dont les questions sont de 3 natures :

- **les questions à 2 propositions**: dans ce cas une seule des 2 propositions est bonne.
  - +1 pour la réponse bonne
  - –1 pour la réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est bonne
  - + 1 pour la réponse bonne
  - –½ pour chaque réponse fausse
- **les questions à 3 propositions** dont 1 seule proposition est fausse
  - + ½ pour chaque réponse bonne
  - –1 pour la réponse fausse

Il s'agit de faire une croix dans les cases de droite en face des propositions.

On peut remarquer que cocher toutes les propositions d'une question revient à ne rien cocher du tout (égal à 0).

Si vous devez raturer une croix, faites-le correctement afin qu'il n'y ait aucune ambiguïté.

N'oubliez pas d'inscrire en en-tête du QCM, votre nom et prénom.

Vous avez droit à **4 points** négatifs sans pénalité.

NOM:	PRENOM:

Une architecture logicielle est un document qui décrit l'expression du besoin sous la forme de spécifications textuelles précises		Q 1.
1	OUI	
2	NON	**X**

Une architecture logicielle est un document qui décrit les composants logiciels et leurs dépendances mutuelles		Q 2.
1	OUI	**X**
2	NON	

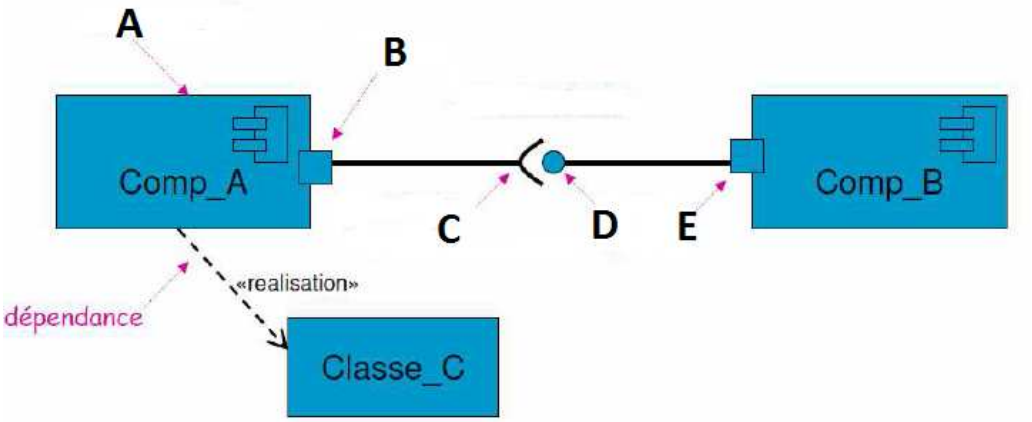
Dans une architecture logicielle, un composant est :		Q 3.
1	une unité de composition logicielle, exposant des interfaces bien spécifiées	**X**
2	une unité de composition logicielle, susceptible d'être déployé de manière indépendante	**X**
3	une unité de composition logicielle spécifique qui ne peut plus se décomposer en d'autres unités de composition logicielle	

Une **Configuration Architecturale** est composée de 3 parties : l' **Architecture Applicative**, l' **Architecture Technique**, et l' **Architecture Physique**.		Q 4.
1	OUI	
2	NON	**X**

L' Architecture Système est l'architecture logicielle d'un système d'exploitation		Q 5.
1	OUI	
2	NON	**X**

Une interface est attachée à un port de communication du composant		Q 6.
1	OUI	**X**
2	NON	

Plusieurs interfaces peuvent être attachées à un même port		Q 7.
1	OUI	**X**
2	NON	

Soit le schéma suivant de description d'un connecteur entre deux composants :		Q 8.
		
avec la légende suivante :		
1	A=composant; B=interface de Comp_A; C=Flot de communication (input) D=Flot de communication (output) E=interface de Comp_B	
2	A=Producteur d'évènement B=File d'évènements produits par Comp_A C=Evènement entrant D=Evènement sortant E=file d'évènements consommés par Comp_B	
3	A=composant; B=port; C=interface requise; D=interface fournie E=port	**X**

En architecture logicielle, un connecteur est toujours une connexion entre deux composants distants		Q 9.
1	OUI	
2	NON	**X**

Un objet « persistant » est un objet créé dans une application orienté objet et rendu susceptible de persister dans une base de données		Q 10.
1	OUI	**X**
2	NON	

La solution technique la plus courante pour faire persister un objet est un SGBDR mais on peut aussi utiliser le XML		Q 11.
1	OUI	**X**
2	NON	

Dans l'approche ORM, le mapping de l'association UML *..* entre deux classes est toujours réalisé par la création d'une table d'association		Q 12.
1	OUI	**X**
2	NON	

Dans une Architecture N-tiers, la couche DAO (Data Access Object) s'intercale entre le tier-IHM et le tier-Métier facilitant pour les clients IHM (Navigateur) l'accès aux données gérées par la couche métier		Q 13.
1	OUI	
2	NON	**X**

Dans une architecture dite "à base de composant" (exemple: J2EE), les composants s'exécutent dans un container. Ce dernier assure :		Q 14.
1	le déploiement des composants sur le tier métier	
2	la communication entre les composants	**X**
3	la localisation et la résolution des dépendances entre composant	**X**

Un modèle d'architecture J2EE est constitué ~~au moins~~ de :		Q 15.
1	2 tiers (Client, Web)	
2	3 tiers (Client, Web, Métier)	**X**
3	4 tiers (Client, Web, Métier, Base de données)	**X**

Un EJB session est un container dans lequel s'exécutent des composants distribués (RMI ou CORBA)		Q 16.
1	OUI	
2	NON	**X**

L'injection de dépendance est un principe de programmation qui permet de réaliser la dépendance entre deux classes en utilisant un fichier de configuration ou des annotations dans le code		Q 17.
1	OUI	**X**
2	NON	

Un EJB container est un environnement dans lequel s'exécutent des EJB session		Q 18.
1	OUI	**X**
2	NON	

Un EJB session est toujours un composant sans états (stateless), les états étant gérés en base de données via un lien de persistence		Q 19.
1	OUI	
2	NON	**X**

Dans une architecture Web Services qui repose sur le protocole SOAP, l'interface de définition d'un service est décrite par :		Q 20.
1	un fichier écrit au standard IDL	
2	un fichier écrit au standard UDDI	
3	un fichier écrit au standard WSDL	**X**

Le protocole SOAP (Simple Object Access Protocol) est un protocole :		Q 21.
1	dont les données échangées peuvent être sérialisées en XML	**X**
2	assurant l'échange d'informations et l'invocation de méthodes distantes (RPC)	**X**
3	assurant l'échange d'informations et l'invocation de méthodes distantes en RMI (Remote Method Invocatoin)	

Pour pouvoir réaliser une communication SOAP, le middleware utilisé génère la souche cliente (stub) et la souche serveur (skelton) permettant l'appel des méthodes d'un service par le client		Q 22.
1	OUI	**X**
2	NON	

Le standard WSDL permet de définir des liaisons de communications (binding) basées sur un autre standard que SOAP		Q 23.
1	OUI	**X**
2	NON	

Dans une architecture Web Services un client identifie les méthodes d'un service en demandant à l'UDDI le WSDL qui décrit l'interface du service		Q 24.
1	OUI	
2	NON	**X**

Un fichier WSDL est un fichier binaire généré par le WebServices et exploité par le client pour savoir comment communiquer avec le serveur		Q 25.
1	OUI	
2	NON	**X**

REST (Representational State Transfer) est un canevas architectural permettant de créer des Web Services		Q 26.
1	OUI	**X**
2	NON	

En REST, les types de requêtes HTTP : GET, POST, UPDATE et DELETE correspondent à la gestion d'une ressource distante (Create, Read, Update et Delete). La création d'une ressource se fait par la requête GET		Q 27.
1	OUI	
2	NON	**X**

En REST, c'est le serveur qui décide le type de retour de la ressource demandée		Q 28.
1	OUI	
2	NON	**X**

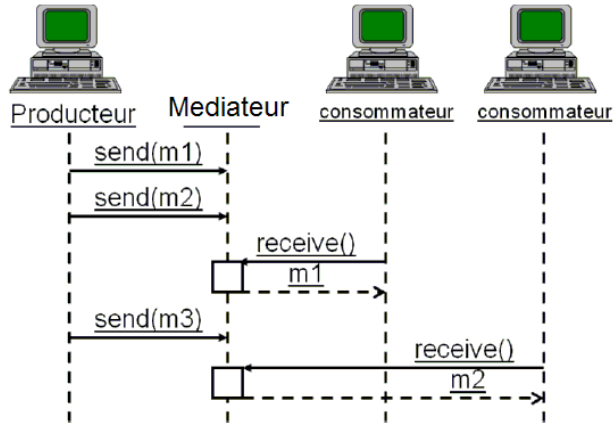
La mise en œuvre d'un Web Service en REST est une opération complexe nécessitant le déploiement sur le serveur d'application de composant CRUD qui gère l'état d'une ressource éventuellement persistant en base de données		Q 29.
1	OUI	
2	NON	**X**

En REST, le serveur d'application est sans état par rapport aux requêtes traitées. Cet état doit être géré par le client.		Q 30.
1	OUI	**X**
2	NON	

Un MOM :		Q 31.
1	est un composant logiciel (Model Orienté Message) qui permet de centraliser l'ensemble des données (Model) d'un système d'information qui sont mises à jour par l'envoi de messages	
2	est une API et des composants dynamiques (Middleware Orienté Message) qui permet certains services d'échanges entre les applications d'un système d'information	**X**

Dans un MOM, les envois de messages sont :		Q 32.
1	toujours synchrone	
2	toujours asynchrone	
3	synchrone ou asynchrone (en fonction du contexte)	**X**

La définition de l'envoi d'un message synchrone entre un producteur et plusieurs consommateurs est :		Q 33.
1	Le producteur envoie un message à un intermédiaire qui lui confirme la prise en compte de la réception de ce message. Puis, l'intermédiaire envoie ce message à tous les consommateurs.	
2	Avant d'envoyer un nouveau message, le producteur attend que le message envoyé ait été consommé par tous les consommateurs.	**X**

En JMS (Java Messaging System), il existe (notamment) deux modes de communication : Queue et Topic.  Ce diagramme de transition correspond au mode :		Q 34.
1	Queue	**X**
2	Topic	

En JMS (Java Messaging System), les producteurs et les consommateurs sont tous des clients d'un JMS Provider		Q 35.
1	OUI	**X**
2	NON	

Fin du QCM

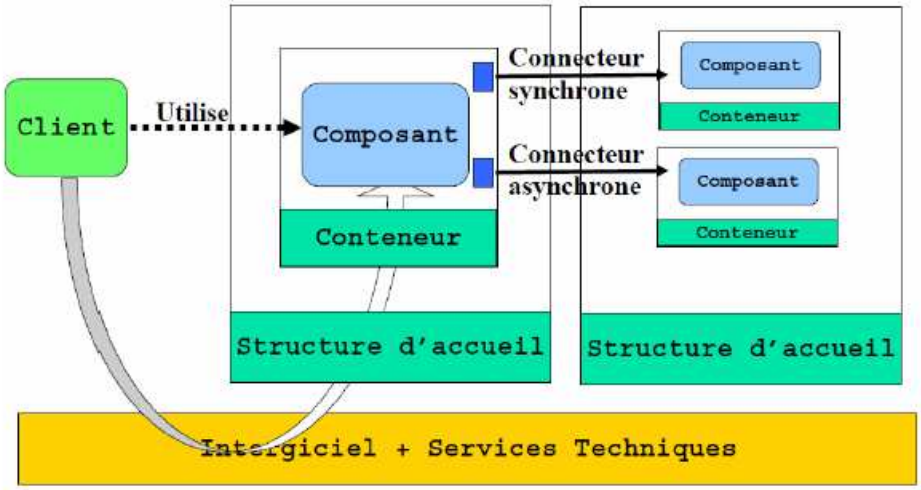
Suite (Tournez la page)

2. Questions libres (15 points)

Chaque question est notée sur 5 points.
*Vous répondez à ces questions sur une **copie vierge double** en mettant bien le numéro de la question, sans oublier votre nom et prénom.*
Vous mettez le QCM dans cette copie vierge double.

QUESTION NUMERO 1

Soit le schéma suivant qui décrit le modèle d'architecture à base de composants (exemple: J2EE) :



Commentez ce schéma tout en mettant en évidence les 2 ou 3 principes forts d'un tel style d'architecture.

Le composant est déployé dans un conteneur lors de la mise en place du système d'information. Le conteneur assure l'exécution du composant. Le conteneur s'exécute dans une structure d'accueil qui exploite les services techniques. Les différents types de conteneur/structure d'accueil sont des éléments logiciels appartenant au framework (exemple J2EE). Ainsi, un point fort est que **l'on concentre l'effort de développement au seul composant**.

Le client utilise des éléments des services techniques pour utiliser les composants. Le deuxième point fort est que **le client ne voit que le composant utilisé** alors que pour communiquer avec lui, il utilise plusieurs couches successives prédéfinis (Intergiciel, Structure d'accueil, Conteneur).

Le deuxième point fort est que **les composants communiquent entre eux à travers des connecteurs**.

Une structure d'accueil peut contenir plusieurs conteneurs.

QUESTION NUMERO 2

Nous avons vu dans le cadre des MOM (Middleware Orientés Message), que JMS (Java Messaging System) est une API JEE répondant au principe d'un MOM.

Expliquez quel est le principe général de JMS en utilisant les termes: JMS Client, JMS Provider, JMS Producer, JMS Consumer, JMS Message.

JMS est un middleware permettant d'échanger des données (appelées aussi évènements) entre des producteurs (JMS Producer) et des consommateurs (JMS Consumer). Les producteurs et les consommateurs sont tous des clients (JMS Client) d'un composant dynamique de JMS, le JMS Provider. Le JMS Provider sert d'intermédiaire de communication entre les producteurs et les consommateurs. Cet intermédiaire gère des canaux d'évènements. Tout consommateur qui est connecté à un canal d'évènement reçoit les évènements des producteurs qui sont connectés à ce même canal.

QUESTION NUMERO 3

Pourquoi le principe d'architecture REST est plus facile de mise en œuvre que celle de J2EE ?

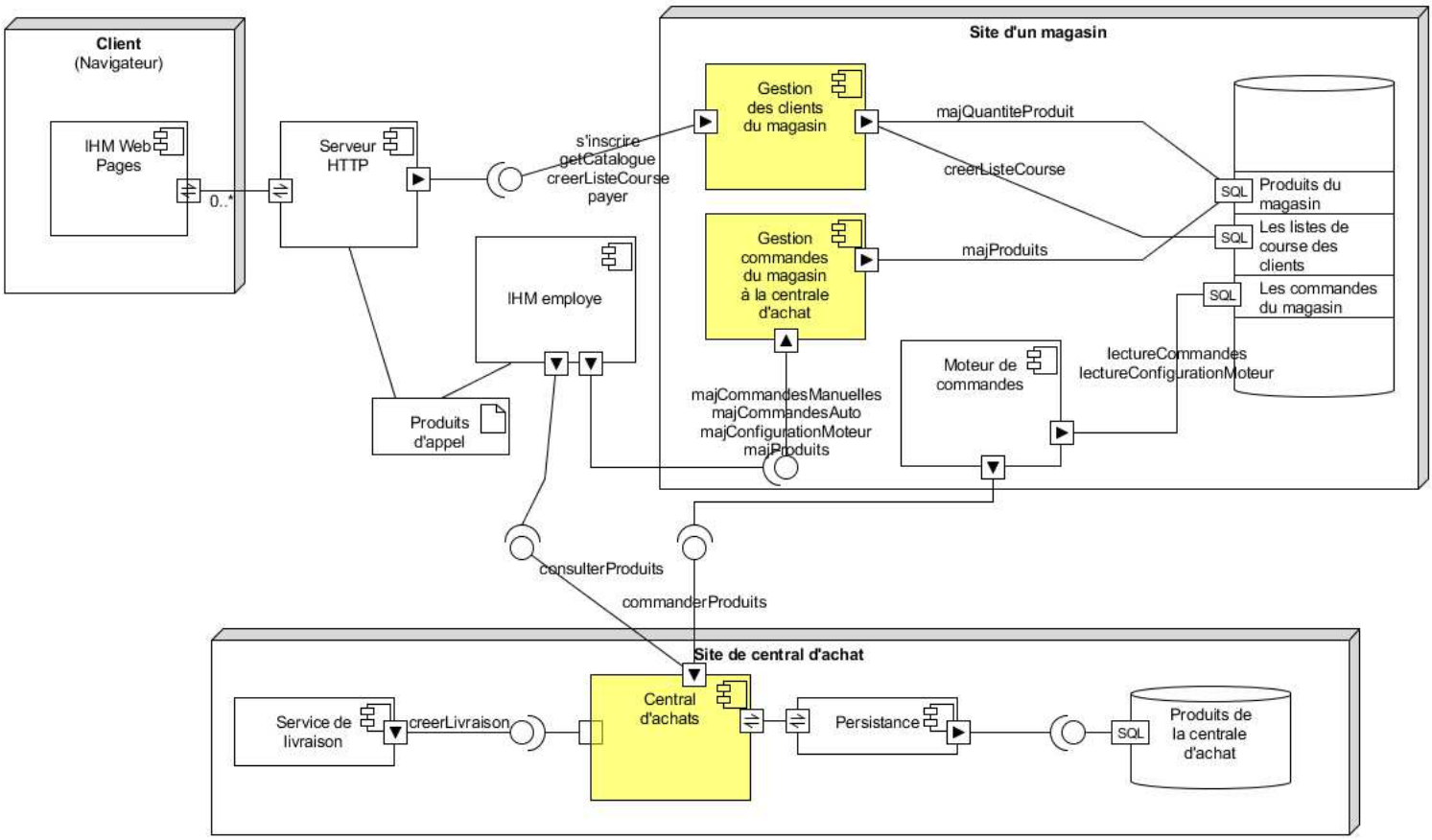
La mise en œuvre de REST est plus facile que celle de J2EE car REST n'utilise pas d'autre standard que ceux utilisés dans le protocole http.
REST exploite des "ressources" du serveur à travers les seules requêtes GET, POST, UPDATE et DELETE. Le serveur retourne une information interprétable par le navigateur.

Fin de la 1ère partie sans document

2ème PARTIE – AVEC DOCUMENT (durée: 1h15)

3. PROBLEME (50 points)

1/ **Le schéma est le suivant** :



[15 points] pour le schéma du tier-IHM et tier-métier du 1er site (dans ma correction : Client + Site d'un magasin)
[5 points] pour le schéma du tier métier du 2ème site (dans ma correction : Site de central d'achat)

[3 points] **Identification des composants qui sont des WebServices**
En jaune les composants qui sont des WebServices.

[13 points] **Commentaires de : Client + Site d'un magasin :**
Avec un navigateur, à travers les pages HTML, un client utilise une IHM qui utilise les services d'un WebService "Gestion des clients du magasin" avec lequel il peut s'inscrire, obtenir le catalogue des produits disponibles, créer sa liste de cours et payer.
Ce Webservice utilise une base de données locale afin d'enregistrer la liste de course du client et mettre à jour la quantité des produits de la liste de course.

Un employé utilise une IHM intranet qui se connecte au WebService "Gestion commandes" pour mettre à jours les commandes manuelles et les commandes automatiques qui sont enregistrés en base de données.
L'IHM se connecte aussi au WebService "Central d'achats" pour consulter le catalogue des produits disponibles gérés par la centrale d'achat.
L'IHM lui permet aussi de configurer le composant "Moteur commandes".

L'IHM permet aussi de mettre à jour les fichiers qui décrivent les produits d'appel qui seront présentés dans la page d'accueil du navigateur.

Le composant "Moteur commandes" est un composant dynamique qui exécute un moteur à un instant programmé (configuration).
Il lit sa configuration dans la BD et lit les produits qu'il doit commander au près de la centrale d'achat.
Il se connecte au WebService "Central d'achat", pour passer sa commande.

[4 points] **Commentaires de : Central d'achat :**

Le WebService "Central d'achat" met à jour la quantité des produits de la base de données locale via un mécanisme de persistance.
Il utilise un composant "Service livraison" qui sort le bon de livraison afin de préparer la livraison.

[10 points] **2/ Description d'un WSDL :**

Je choisis le WebService "Central d'achats".
Ce WebService définit 2 ports de communication. Un premier port réalise l'interface **Consultation** et le deuxième l'interface **Commander**.

L'interface **Consultation** définit la méthode **consulterProduits** qui retourne la liste de tous les produits disponibles de la centrale d'achat.

L'interface **Commander** définit la méthode **commanderProduits** qui prend en entrée l'id du magasin et une liste de produits à commander. Il retourne la liste des produits qui ne sont plus disponibles car entre le temps de préparer la commande et son envoi il peut se passer du temps (la commande est réalisée le dimanche).

```
<wsdl : definitions name= ”Catalogue”
        targetNamespace=”http://www.entreprise.org”
        xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
        xmlns:wsdl=”http://schemas.xmlsoap.org/wsdl”>

<wsdl:types>
        <xsd:schema targetNamespace=”http://www.entreprise.org/types”>
                <xsd:complexType name=“produit”>
                        <xsd:sequence>
                                <xsd:element name=”ID” type=”xsd:integer”
                                        minOccurs=”1” maxOccurs=”1”/>
                                <xsd:element name=”nom” type=”xsd:string”
                                        minOccurs=”1” maxOccurs=”1”/>
                                <xsd:element name=”descriptif” type=”xsd:string”
                                        minOccurs=”1” maxOccurs=”1”/>
                                <xsd:element name=”prix” type=”xsd:decimal”
                                        minOccurs=”1” maxOccurs=”1”/>
                        </xsd:sequence>
                </xsd:complexType>
                <xsd:complexType name=“listeProduit”>
                        <xsd:sequence>
                                <xsd:element name=”item” type=”tns.produit”
                                        minOccurs=”0” maxOccurs=”unbounded”/>
                                </xsd:sequence>
                </xsd:complexType>

        </xsd:schema>
```

```
</wsdl:types>

<wsdl:message name=consulterProduitsRequete/>
</wsdl:message name=consulterProduitsReponse >
        <part name=”Resultat” type=”tns:listeProduit”/>
</wsdl:message>

<wsdl:message name=commanderProduitsRequete>
        <part name=”IdMagasin” type=”tns:string”/>
        <part name=”Commande” type=”tns:listeProduit”/>
</wsdl:message>

</wsdl:message name=commanderProduitsReponse >
        <part name=”Resultat” type=”tns:listeProduit”/>
</wsdl:message>



< wsdl:portType name=”consultation”>
        <wsdl:operation name=”consulterProduit”>
                <wsdl:input message=” consulterProduitRequete”/>
                <wsdl:output name=” consulterProduitReponse”>
        </wsdl:operation>
</wsdl :portType>

< wsdl:portType name=”commande”>
        <wsdl:operation name=”commanderProduits”>
                <wsdl:input message=” commanderProduitsRequete”/>
                <wsdl:output name=” commanderProduitsReponse”>
        </wsdl:operation>
</wsdl :portType>


<wsdl:binding name=”consultationBinding” type=”tns:consultation>
        <soap:binding transport=”http://schemas.xmlsoap.org/soap/http” style=“rpc”/>
                <wsdl:operation name=” consulterProduit”>
                        <soap:operation soapAction= “ …….”/>
                        <wsdl:input>
                                <soap:body use=”literal”
                                        namespace=”http://catalogueWebService.com”/>
                        </wsdl:input>
                        <wsdl:output>
                                <soap:body use=”literal”
                                        namespace=”http://catalogueWebService.com”/>
                        </wsdl:output>
                </wsdl:operation>
</wsdl:binding>

<wsdl:binding name=”commandeBinding” type=”tns:commande>
        <soap:binding transport=”http://schemas.xmlsoap.org/soap/http” style=“rpc”/>
                <wsdl:operation name=” commanderProduits”>
                        <soap:operation soapAction= “ …….”/>
                        <wsdl:input>
                                <soap:body use=”literal”
                                        namespace=”http://catalogueWebService.com”/>
                        </wsdl:input>
                        <wsdl:output>
                                <soap:body use=”literal”
                                        namespace=”http://catalogueWebService.com”/>
                        </wsdl:output>
```

```
                </wsdl:operation>
</wsdl:binding>


<wsdl:service name=”CentralAchats”>
        <wsdl:port name=”commander” binding=tns:commanderBinding >
                <soap:address location=”http://soap.entreprise.org/catalogue/soap”
        </wsdl:port>
        <wsdl:port name=”consulter” binding=tns:consulterBinding >
                <soap:address location=”http://soap.entreprise.org/catalogue/soap”
        </wsdl:port>
</wsdl:service>

</definitions>
```