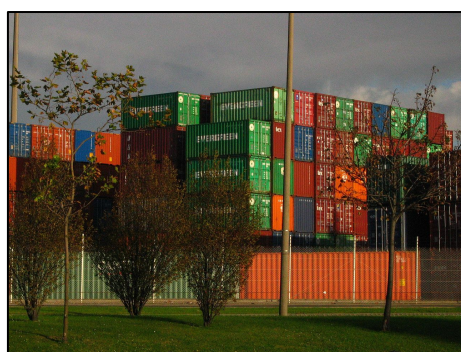
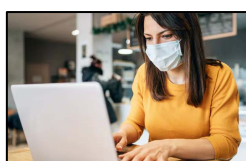


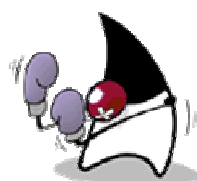
## Laboratoire de Réseaux et technologie Internet (TCP-UDP/IP & HTTP en C/C++/Java) :

3<sup>ème</sup> Informatique de gestion  
& 3<sup>ème</sup> Informatique et systèmes  
opt. Informatique industrielle et Réseaux-télécommunications  
2020-2021



## Inpres PFM (Plate-Forme Multimodale)

Claude Vilvens, Christophe Charlet, Sébastien Calmant  
(Network programming team)



## 1. **Préambule**

L'Unité d'Enseignement "**Programmation réseaux, web et mobiles**" (10 ECTS - 135h) comporte des Activités d'apprentissage :

- ◆ AA: Réseaux et technologies Internet (dans toutes les options);
- ◆ AA: Programmation.Net (dans toutes les options);
- ◆ AA: Technologie de l'e-commerce et mobiles (informatique de gestion seulement);
- ◆ AA: Compléments de programmation réseaux (informatique réseaux-télécoms seulement).

Les travaux de programmation réseaux présentés ici constituent la description technique des travaux de laboratoire de l'AA "**Réseaux et technologie Internet**". Il s'agit ici de maîtriser les divers paradigmes de programmation réseau TCP/IP et HTTP dans les environnements UNIX et Windows, en utilisant les langages C/C++ et Java. Ces travaux ont été conçus de manière à pouvoir collaborer avec les travaux de laboratoire des AA "**Compléments programmation réseaux**" (3<sup>ème</sup> informatique réseaux-télécoms) et "**Technologies du e-commerce**" (3<sup>ème</sup> informatique de gestion); certains points correspondants sont donc déjà vaguement évoqués ici.

Les développements C/C++ UNIX sont sensés se faire avec les outils de développement disponibles sur la machine Sunray; on pourra aussi utiliser une machine virtuelle Sun Solaris ou Linux. Cependant, Code::Blocks sur les PCs peut vous permettre de développer du code en dehors de ces machines. Les développements Java sont à réaliser avec **NetBeans 8.\***. Le serveur Web avec moteur à servlets/JSP utilisé est **Tomcat 7\*/8\*** sous Windows (PC) et/ou Unix (machines U2 ou INXS). La gestion des fichiers élémentaires se fera au moyen de fichiers à enregistrements classiques, de fichiers textes ou de fichiers CSV (dans le contexte C/C++) ou properties (dans le contexte Java). La gestion des bases de données intervenant dans cet énoncé se fera avec le SGBD **MySQL**, interfacé avec la ligne de commande ou, de manière plus attrayante, avec des outils comme **MySQL Workbench** ou **Toad for MySQL**.

Les travaux peuvent être réalisés

- ◆ soit par pour **une équipe de deux étudiants** qui devront donc se coordonner intelligemment et se faire confiance, sachant qu'il faut être capable d'expliquer l'ensemble du travail (pas seulement les éléments dont on est l'auteur);
- ◆ soit par un **étudiant travaillant seul** (les avantages et inconvénients d'un travail par deux s'échangent : on a moins de problèmes quand il faut s'arranger avec soi-même et on ne perd pas de temps en coordination).

## 2. Règles d'évaluation

Comme on sait, la note finale pour l'UE considérée se calcule par une moyenne des notes des AA constitutives, sachant que le seul cas de réussite automatique d'une UE est une note de 10/20 minimum dans chacune des AAs.

Pour ce qui concerne l'évaluation de l'AA "Réseaux et technologie Internet", voici les règles de cotation utilisées par les enseignants de l'équipe responsable de cette AA.

1) L'évaluation établissant la note de l'AA "Réseaux et technologie Internet" est réalisée de la manière suivante :

- ♦ théorie : un examen écrit en janvier 2021 (sur base d'une liste de points de théorie à développer fournis au fur et à mesure de l'évolution du cours théorique) et coté sur 20;
- ♦ laboratoire : 4 évaluations pondérées selon leur importance (les 3 premières en évaluation continue, non remédiables car visant à acquérir des notions de base, la dernière remédiable en 2<sup>ème</sup> session), chacune notée sur un total établi en fonction de l'importance des notions à assimiler et de la charge de travail correspondante; la moyenne de ces notes fournit une note de laboratoire sur 20;
- ♦ **note finale : moyenne de la note de théorie (poids de 50%) et de la note de laboratoire (poids de 50%).**

Dans ces conditions, *il est clair qu'une note de théorie beaucoup trop basse (du type 5/20 ou moins encore) ne peut que conduire à l'échec de l'AA considérée.*

Cette procédure est d'application tant en 1<sup>ère</sup> qu'en 2<sup>ème</sup> session.

2) Dans le cas où les travaux sont présentés par une équipe de deux étudiants, chacun d'entre eux doit être capable d'expliquer et de justifier l'intégralité du travail sans de longues recherches dans le code de l'application proposée (pas seulement les parties du travail sur lesquelles il aurait plus particulièrement travaillé).

3) Dans tous les cas, tout étudiant doit être capable d'expliquer de manière générale (donc, sans entrer dans les détails) les notions et concepts théoriques qu'il manipule dans ses travaux (par exemple: socket, machine à états de TCP, signature électronique, certificat, etc).

4) En 2<sup>ème</sup> session, un **report de note** est possible pour **des notes supérieures ou égales à 10/20** en ce qui concerne :

- ♦ la note de théorie;
- ♦ les notes de laboratoire de l'évaluation 4 (évaluation à l'examen).

Les évaluations de théorie et du laboratoire 4 ayant des **notes inférieures à 10/20** sont donc **à représenter dans leur intégralité** (le refus de représenter une évaluation complète de laboratoire entraîne automatiquement la cote de 0).

**Les notes de laboratoire des évaluations 1, 2 et 3 ne sont pas remédiables** : comme elles ont pour but de faire acquérir des techniques élémentaires, il n'a plus de sens de tester à nouveau ces acquis en 2<sup>ème</sup> session et elles resteront donc à la valeur acquise lors de l'évaluation de 1<sup>ère</sup> session.

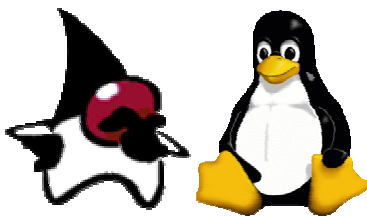
### **3. Agenda des évaluations**

Pour chaque évaluation, le délai est à respecter impérativement.

<b>Evaluation</b>	<b>Evaluation continue : semaine d'évaluation</b>	<b>Pondération dans la cote de laboratoire</b>	<b>Remédiable en 2<sup>ème</sup> session</b>
<b>Evaluation 1</b> : client-serveur multithread TCP en C/C++	5/10/2020-9/10/2020	20	non
<b>Evaluation 2</b> : JDBC + client-serveur multithread TCP en Java	26/10/2020-31/10/2020	30	non
<b>Evaluation 3</b> : programmation Web Java classique	23/11/2020-27/11/2020	30	non
<b>Evaluation 4</b> : <ul style="list-style-type: none"><li>♦ architecture complète fonctionnelle</li><li>♦ client-serveur sécurisé en Java</li><li>♦ communications réseaux C/C++-Java</li><li>♦ client-serveur UDP en C/C++ et Java</li></ul>	Examen de laboratoire de janvier 2021	120	oui

**Remarque importante** : Pour rappel, lors de chaque évaluation, chaque étudiant est sensé connaître les bases théoriques qui lui ont permis de réaliser les développements proposés. Dans le cas contraire, on sera amené à considérer qu'il a développé sans comprendre ce qu'il faisait ou, pire, que le travail proposé a été récupéré ailleurs (plagiat pur et simple) ... ce qui conduit automatiquement à une note insuffisante.

### **4. Avertissements préalables**



1) Dans ce qui suivra, un certain nombre de points ont été simplifiés par rapport à la réalité. **Certains points ont également été volontairement laissés dans le vague**: il vous appartient d'élaborer vous-mêmes les éléments qui ne sont pas explicitement décrits dans l'énoncé. Cependant, pour vous éviter de vous fourvoyer dans une impasse ou perdre votre temps à des options de peu d'intérêt, il vous est vivement recommandé de prendre conseil au près de l'un de vos professeurs concernés par le projet.

2) Le contenu des évaluations a été déterminé en fonction de l'avancement du cours théorique ET des besoins des autres cours de 3<sup>ème</sup> bachelier, avec lesquels nous nous sommes synchronisés dans la mesure du possible. Sans ces contraintes, le schéma de développement eût été différent.

3) Autre chose : une condition sine-qua-non de réussite est le traitement systématique et raisonné des erreurs courantes (codes de retour, errno, exceptions).

4) Le nom de l'UE et de l'AA comporte le mot "**réseaux**": si travailler au départ en **localhost** est légitime, **il est par contre impérieux de présenter un dossier final qui fonctionne en réseau effectif** (donc client et serveur sur deux machines distinctes).

Une plage de 10 ports TCP-UDP vous est attribuée sur les machines Unix et Linux. Il vous est demandé de la respecter pour des raisons évidentes ...



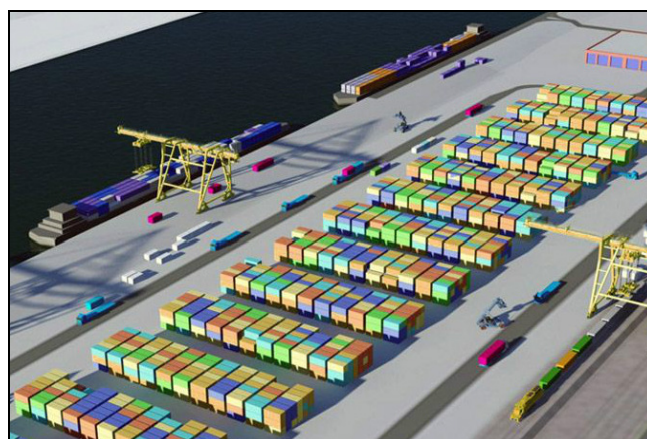
## **5. Le contexte général**

Il s'agit ici de développer diverses applications permettant de gérer une plate-forme multimodale (PFM) située à Frog Gulch, en Boursoulavie (à l'image de celle que l'on trouve par exemple, et réellement, à Oupeye (province de Liège) depuis 2015).



Une PFM est donc un centre logistique qui concentre des marchandises de tout type dans des containers : ceux-ci y sont amenés par un moyen de transport (routier, fluvial ou ferroviaire) pour être emmené vers une destination donnée par un autre moyen. Il s'agit donc d'une plaque-tournante de circulation de marchandises.

Dans notre cas, comme les nuisances pour les riverains risquent d'être importantes sans un suivi rigoureux, une vaste zone verte tampon a été prévue; elle est même envisagée partiellement comme réserve naturelle et partiellement comme parc de loisirs "verts". Dans les deux cas, le public devra réserver sa visite car on souhaite éviter une présence humaine trop forte et donc réguler celle-ci afin de privilégier une certaine vie sauvage.



La plate-forme considérée ici se nomme Inpres-PFM et comporte donc :

- ◆ un réseau routier approprié avec accès aux voies rapides (ici, l'autoroute E007 et la nationale N69);
- ◆ une ligne de chemin de fer (gérée par la SNCFB) avec quais de chargement et voies de garage;

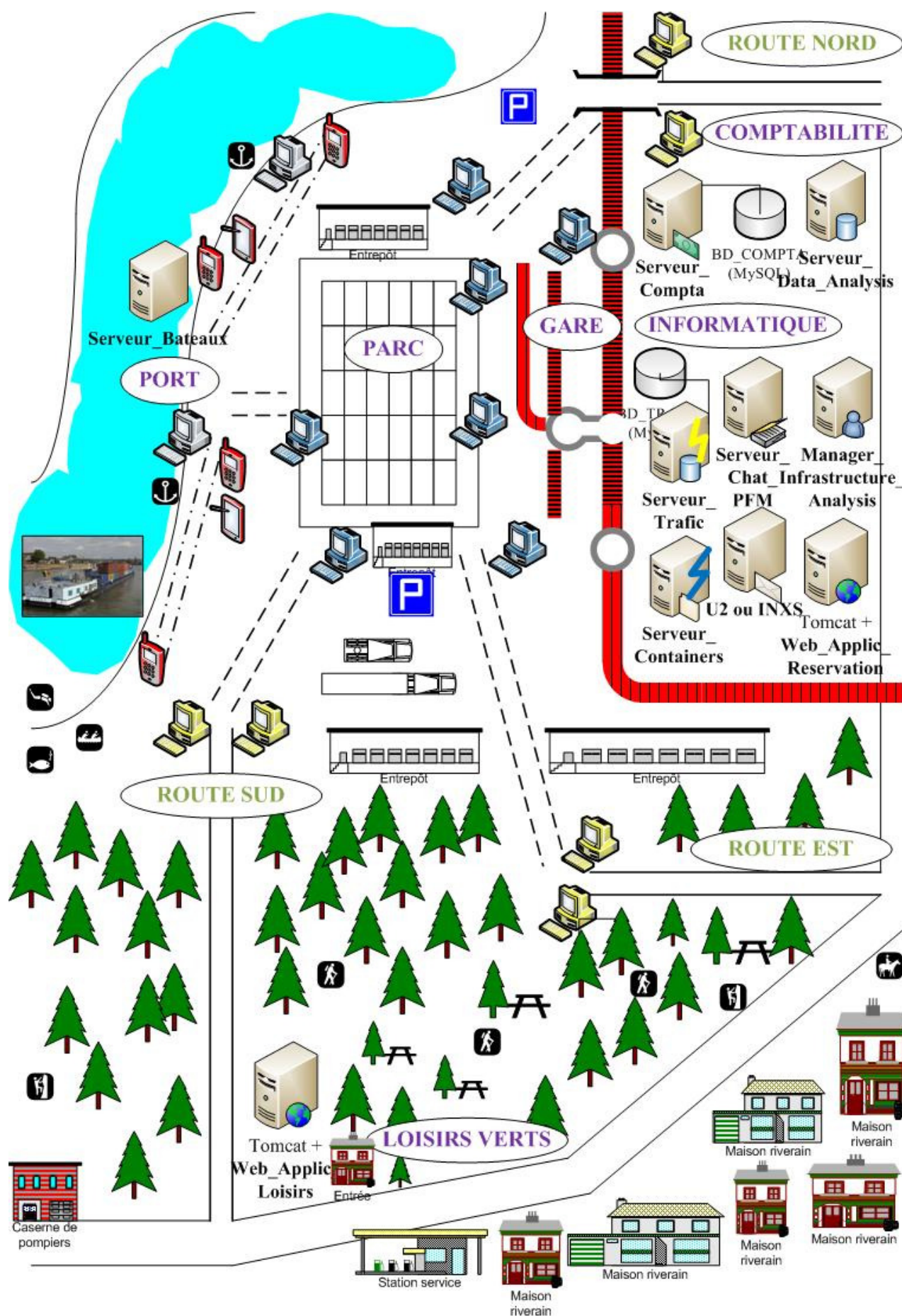
◆ un accès à une voie navigable (ici, le fleuve Brahmaproutprout) avec infrastructures portuaires.

## **6. Le cahier des charges**

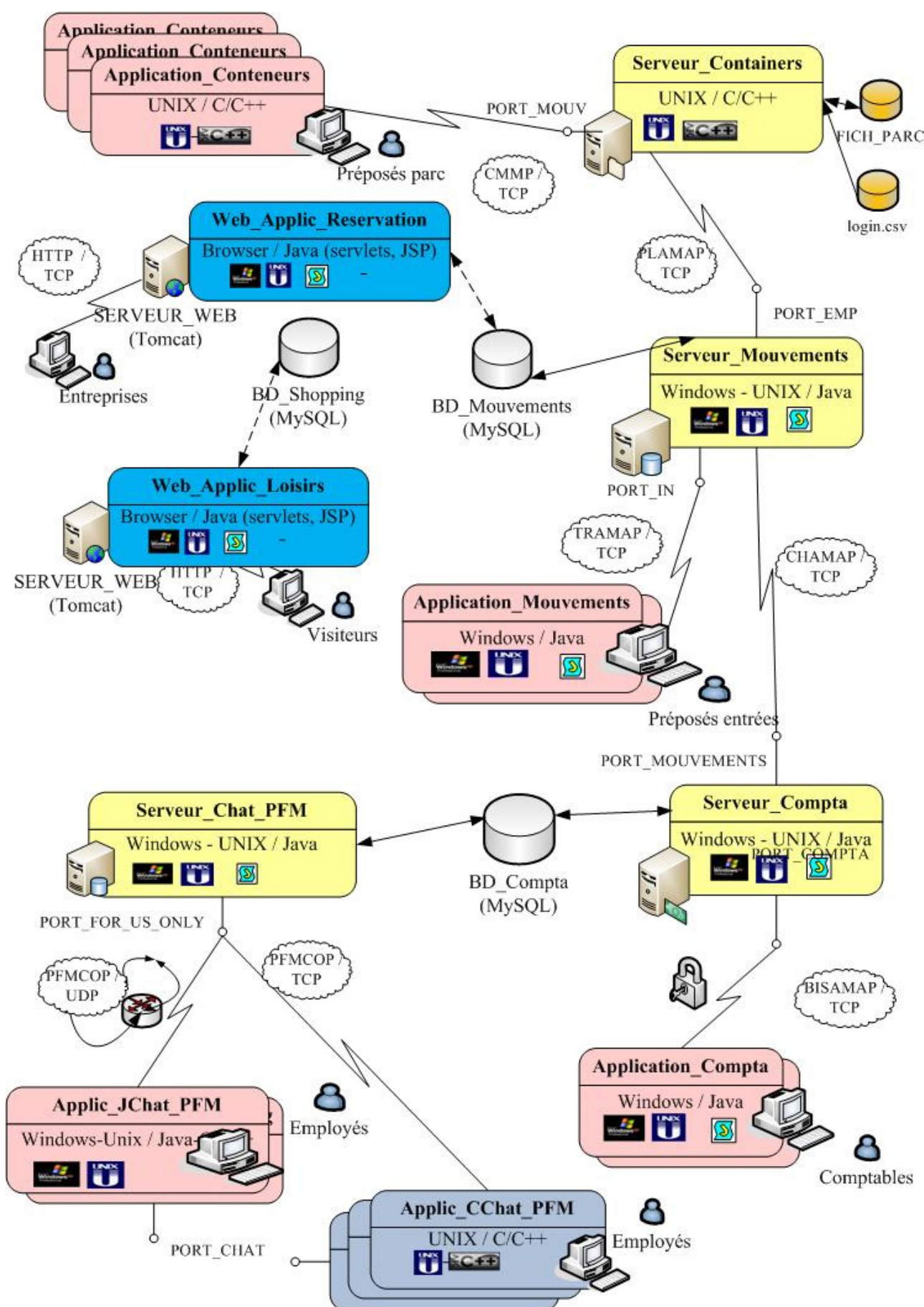
Du point de vue informatique, la solution informatique à construire consiste à mettre en place l'infrastructure globale suivante :

- ◆ un système de gestion du Mouvements des camions et containers : arrivée et départ des containers, stockage intermédiaire, traçage en temps réel, etc; ce système est assuré par un serveur **Serveur\_Mouvements** de type Java/Windows-Unix qui centralise l'ensemble du Mouvements et donc notamment les positions des containers dans le parc; une base de données MySQL BD\_MOUVEMENTS est utilisée par ce serveur;
- ◆ un système de gestion du parc de containers (dépôt à l'endroit prévu ? enlevage pour aller sur quel transporteur ?) avec un serveur de production **Serveur\_Containers** de type C/Unix qui centralise l'ensemble des dépôts et déplacements des containers du parc; pour des raisons techniques d'accès par des dispositifs à software de niveau proche des machines électromécaniques, les données sont gérées au moyen de simples fichiers à enregistrements;
- ◆ une application Web **Web\_Applic\_Reservation**, basée sur la technologie servlets/JSP qui permet aux entreprises de réserver des emplacements et des transferts pour leurs containers;
- ◆ tout ce qui concerne les facturations de services, les locations d'emplacements, les amarrages, etc, mais aussi les salaires du personnel est du ressort du système dont le serveur multithread **Serveur\_Compta** (de type Java/Windows-Unix) est le centre; une base de données MySQL BD\_COMPTA est utilisée par ce serveur; ici, la sécurité logicielle (implémentée de manière propriétaire avec les outils cryptographiques classiques) devient impérative;
- ◆ une application Web **Web\_Applic\_Loisirs**, basée sur des technologies servlets/JSP qui permet aux particuliers de réserver des visites au parc de loisirs verts et/ou à la réserve naturelle, ou simplement d'acheter des guides natures et autres produits en relation (peluches d'animaux, accessoires de bureau "nature", etc);
- ◆ un système de chat interne, mis en place dans un esprit de maintien d'un esprit d'entreprise entre tous les membres de la société – (presque) tout peut y être dit; un serveur **Serveur\_Chat\_PFM** Java/Windows-Unix vérifie l'appartenance à la société Inpres-PFM et permet aux utilisateurs d'une application **Applic\_JChat\_PFM** (Java) ou **Applic\_CChat\_PFM** (C/C++) de se joindre au groupe de discussion de la société;
- ◆ tout ce qui concerne les mouvements de bateaux réclame un maximum de souplesse et est du ressort du système basé sur le serveur multithread **Serveur\_Bateaux\*** (de type Java/Windows-Unix); ses clients sont non seulement des applications installées mais aussi des applications mobiles\*; de plus, il gère des fichiers XML\* utilisés par le service "Maintenance" (le service technique de gestion des infrastructures du port).
- ◆ le service "Informatique" utilise encore deux autres serveurs : un serveur de mails interne (baptisé **U2\*** avec un serveur clone de secours **INXS\*** – ces informaticiens sont très "pop/rock") et un système d'analyse de l'infrastructure **Manager\_Infrastructure\_Analysis\*** ;
- ◆ le service Comptabilité utilise un **Serveur\_Data\_Analysis\*** servant à l'étude des données de l'entreprise (data mining).

Schématiquement, l'environnement général est décrit par le premier schéma suivant, tandis que les implémentations informatiques demandées dans ce laboratoire le sont dans le second :







## 7. Un scénario exemple d'utilisation de l'infrastructure

On remarquera au préalable que, vu la topologie du site, un camion ne peut parvenir au parc de containers qu'après être passé au préalable par un des points d'entrée où se trouvent des applications clientes du Serveur\_Mouvements. Tout camion qui atteint le parc est donc déjà "attendu" par les applications clientes du Serveur\_Containers.



Considérons à titre d'exemple l'entreprise YABBBB (Y A Bon Biscuit du Bon Dieu) qui veut envoyer par bateau un container de ses produits au centre de distribution de Lourdes.

1) Un responsable de la société effectue le 15/9/2020 une réservation par Web d'un emplacement dans le parc pour le 17/9/2020 (**Web\_Applic\_Reservation**) ; il reçoit en retour un numéro de réservation E2020091000237 et pour information l'emplacement prévu (17,19). Il donne cette information au conducteur du camion (immatriculé 1-VIL-007) qui va transporter le container YABB-CHARL-A1B2C3.

2) Le camion se présente au jour dit à un poste d'entrée "route" : il produit son numéro de réservation E2020091000237 et se voit délivrer un ticket avec son numéro d'emplacement : en définitive, ce sera plutôt le (5,19) (**Serveur\_Mouvements**).

3) Le camion se rend ensuite à l'entrée du parc à containers qui se trouve au bout de la route qu'il a empruntée et s'y fait enregistrer (**Serveur\_Containers**); le container est déchargé, pesé (15.6 tonnes) et placé au bon endroit.



Un record du fichier FICH\_PARC ➔ un tuple de la table Parcs a été créé :

- ◆ coordonnées (x,y) de l'emplacement : 5,19
- ◆ identifiant du container: YABB-CHARL-A1B2C3
- ◆ flag l'état de l'emplacement : 2
- ◆ la date éventuelle de réservation (sous forme de chaîne de caractères); 15/9/2020
- ◆ la date d'arrivée (sous forme de chaîne de caractères) : 17/9/2020
- ◆ le poids (contenant + contenu) : 15.6
- ◆ la destination du container : Lourdes
- ◆ le fait qu'il atteindra cette destination par Bateau ou par Train : Bateau

Un tuple a été créé dans la table Mouvements de BD\_MOUVEMENTS et les renseignements de dépôt y sont déjà mémorisés :

- \* identifiant mouvement : M20200915200004
- \* identifiant container : YABB-CHARL-A1B2C3
- \* identifiant transporteur entrant : 1-VIL-007
- \* date arrivée : 17/9/2020
- \* identifiant transporteur sortant : NULL
- \* poids (contenant + contenu) : 15.6
- \* date départ : NULL
- \* destination : Lourdes

<le temps passe ...>

4) Le bateau "Plus près de toi mon Dieu", à destination de Lourdes, arrive au port de la plate-forme le 19/9/2020. Le capitaine se fait enregistrer à l'un des points d'entrée "bateau" du parc (**Serveur\_Containers**) et la liste des containers destinés à ce bateau est établie (**Serveur\_Mouvements**).



5) Un responsable du parc à containers prend connaissance de cette liste (**Serveur\_Containers**) et fait transporter les containers visés vers un quai de chargement puis sur le bateau. Quand tout est terminé et que le bateau part, le responsable le signale (**Serveur\_Containers** et **Serveur\_Mouvements**).

Le tuple de la table Mouvements de BD\_MOUVEMENTS est complété.

- \* identifiant mouvement : M20200915200004
- \* identifiant container : YABB-CHARL-A1B2C3
- \* identifiant transporteur entrant : 1-VIL-007
- \* date arrivée : 17/9/2020
- \* identifiant transporteur sortant : B / Plus près de toi mon Dieu
- \* poids (contenant + contenu) : 15.6
- \* date départ : 19/9/2020
- \* destination : Lourdes

6) La facture est générée automatiquement (**Serveur\_Compta**). Un comptable la validera (ou pas) et l'enverra à la société cliente par la voie la plus appropriée.

**Nous ne considérerons dans ce dossier que les containers apportés par camions et enlevé par bateau ou par train.**

Mais bien sûr, le projet complet devrait envisager d'autres scénarios.

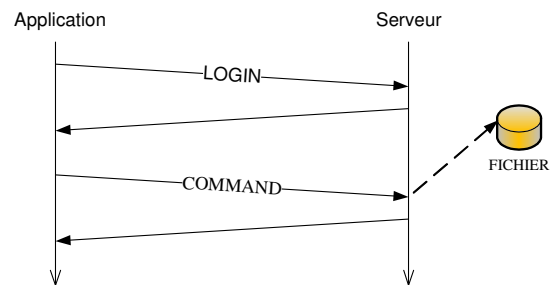
## Les travaux de l'évaluation 1 : client-serveur multithread TCP en C/C++

### Compétences développées :

- ◆ Maîtriser l'implémentation du modèle client-serveur avec sockets TCP en C/C++;
- ◆ Concevoir une librairie de fonctions/classes utiles dans un but d'utilisation simplifiée et réutilisable dans le développement;
- ◆ Développer des fonctions/classes génériques (dissimulant suffisamment leur fonctionnement interne pour que leur utilisation ne souffre pas d'un changement d'implémentation).

### Dossier attendu :

1. tableau des commandes et diagramme du protocole CMMP - du type ci-contre;
2. code C/C+ du serveur Serveur\_Containers
3. vues des trames échangées par un sniffer lors du début des connexions (états TCP).



## 8. Serveur Containers et Application Containers

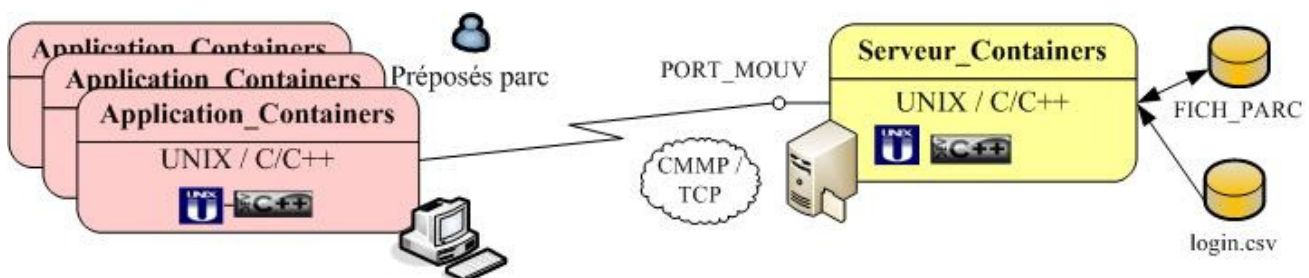
### 8.1 La gestion du parc de containers

Le **Serveur\_Containers** a donc pour rôle de gérer le parc de containers : carte du parc, mouvements, etc. Nous allons développer ce serveur (lui et ses applications clientes) en ignorant provisoirement le **Serveur\_Mouvements** (bien sûr, nous établirons une vraie communication entre les deux serveurs plus loin).

L'occupation du parc est provisoirement enregistrée dans un fichier classique à enregistrements FICH\_PARC (dans la suite, les informations de ce fichier seront transférées dans une table de BD\_MOUVEMENTS). Un record contient simplement :

- ◆ les coordonnées (x,y) de l'emplacement;
- ◆ l'identifiant du container si l'emplacement est occupé ou en passe de l'être;
- ◆ un flag sur l'état de l'emplacement indiquant 0 (libre), 1 (promis), 2 (occupé);
- ◆ la date éventuelle de réservation (sous forme de chaîne de caractères);
- ◆ la date d'arrivée (sous forme de chaîne de caractères);
- ◆ le poids (contenant + contenu);
- ◆ la destination du container;
- ◆ le fait qu'il atteindra cette destination par Bateau ou par Train.

Le serveur est un serveur multithread C/Unix en modèle pool de threads. Il est chargé de répondre aux requêtes provenant de **Application\_Containers** (C/C++) utilisée par les préposés installés aux diverses entrées du parc. Le serveur attend ce type de requête sur le PORT\_MOUV. Il utilise le protocole applicatif (basé TCP) **CMMP** (Container Move Management Protocol).





A cet état d'avancement du projet, nous ne considérerons que les containers entrant par route et sortant par bateau ou par train. Les commandes de CMMP, somme toute très prévisibles vu les scénarios exposés ci-dessus, sont donc :

protocole <b>CMMP</b> - PORT_MOUV		
application cliente : <b>Application_Containers</b>		
commande	sémantique	réponse éventuelle
LOGIN	Un préposé veut se connecter <i>paramètres</i> : nom et mot de passe	oui (validation sur base d'un simple fichier <b>login.csv</b> ) + attente d'une autre commande ou non + cause
INPUT-TRUCK	Arrivée d'un camion apportant un container (pour simplifier: on supposera qu'un camion n'apporte qu'un seul container) <i>paramètres</i> : numéro d'immatriculation du camion + identifiant du container qu'il transporte	oui + emplacement (x,y) qui a été attribué à ce container + enregistrement dans FICH_PARC + attente de la commande INPUT-DONE; ** <i>dans la suite</i> (évaluation 4): envoi de la requête GET_XY à Serveur_Mouvements pour obtenir l'emplacement (x,y) - ce sera donc le serveur Mouvements qui décidera de l'emplacement à attribuer à un container; en attendant que ce serveur existe, Serveur_Containers le génère de manière aléatoire ** ou non + pourquoi (pas d'emplacements libres)
INPUT-DONE	Signale que le container est en place et qu'il a été pesé <i>paramètres</i> : OK + poids de l'ensemble du container (l'application se poursuit sur intervention manuelle de l'opérateur après réception de l'ACK du serveur) ou KO (l'application ferme la connexion et se termine après réception de l'ACK)	oui + enregistrement dans FICH_PARC *** <i>dans la suite</i> (évaluation 4), le serveur enverra une requête SEND_WEIGHT à Serveur_Mouvements) ou non + pourquoi (ex: container non conforme, trop grand, etc)
OUTPUT-READY	Signale qu'un train ou un bateau est disponible pour le chargement de containers <i>paramètres</i> : identifiant train ou bateau, destination, capacité maximale (en containers)	oui + la liste des containers en attente pour cette destination (d'après le fichier FICH_PARC) + attente de la commande OUTPUT-ONE; ** <i>dans la suite</i> (évaluation 2), envoi de la requête GET_LIST à Serveur_Mouvements pour obtenir la liste des emplacements ** ou non + pourquoi (par exemple : aucun container pour cette destination)

OUTPUT-ONE	Signale qu'un container bien précis a été chargé (les containers sont choisis en fonction de la date d'arrivée - ceux qui ont les plus attendus d'abord) <i>paramètres</i> : identifiant du container	oui + mise à jour dans FICH_PARC ou non (container inconnu)
OUTPUT-DONE	Signale que le maximum possible de containers a été chargé <i>paramètres</i> : identifiant train ou bateau, nombre de containers chargé	oui; *** <b><i>dans la suite</i></b> (évaluation 2), <i>envoi de la requête</i> SIGNAL_DEP à Serveur_Mouvements pour signaler la fin des opérations ou non (incohérence détectée)
LOGOUT	Un préposé veut se déconnecter <i>paramètres</i> : nom et mot de passe	oui + session fermée ou non + cause

## 8.2 Quelques conseils méthodologiques pour le développement de CMMP

1) Il faut bien réaliser que la situation gérée ici par Serveur\_Containers est TEMPORAIRE : dès que le serveur Serveur\_Mouvements sera opérationnel, une communication devra être établie entre ces deux serveurs (notamment pour la désignation des emplacements).

**Il conviendra donc de développer Serveur\_Containers en ne perdant pas de vue que des modifications devront être apportées ultérieurement (évaluation 2) avec le minimum de réécriture de code.**

2) Il faut tout d'abord choisir la manière d'implémenter les requêtes et les réponses des protocoles CMMP et plusieurs possibilités sont envisageables pour écrire les trames;

- uniquement par chaîne de caractères contenant des séparateurs pour isoler les différents paramètres;
- sous forme de structures, avec des champs suffisamment précis pour permettre au serveur d'identifier la requête et de la satisfaire si possible;
- un mélange des deux : une chaîne pour déterminer la requête, une structure pour les données de la requête;
- fragmenter en plusieurs trames chaînées dans le cas d'une liste à transmettre.

On veillera à utiliser des constantes (#DEFINE et fichiers \*.h) et pas des nombres ou des caractères explicites.

3) On peut ensuite construire un squelette de serveur multithread et y intégrer les appels de base des primitives des sockets. Mais il faudra très vite (sous peine de réécritures qui feraient perdre du temps) remplacer ces appels par les appels correspondants d'une **bibliothèque (librairie)** facilitant la manipulation des instructions réseaux. Selon ses goûts, il s'agira

- soit d'une bibliothèque C de fonctions réseaux TCP/IP;
- soit d'une bibliothèque C++ implémentant une *hiérarchie de classes* C++ utiles pour la programmation réseau TCP/IP : par exemple, Socket, SocketClient et SocketServeur, éventuellement (mais cela devient très(trop) ambitieux pour le temps dont on dispose), si cela est vraiment estimé utile, flux réseaux d'entrée et de sortie NetworkStreamBase, ONetworkStream et INetworkStream).

Dans les deux cas, un mécanisme d'erreur robuste sera mis au point.

4) Quelques remarques s'imposent :

4.1) Une remarque pleine de bon sens ;-): une fonction (ou une méthode) de bibliothèque ne présente d'intérêt que si elle est plus facile à utiliser que la (les) fonction(s) qu'elle remplace ...

4.2) Une deuxième remarque pleine de bon sens ;-): une fonction (ou une méthode) de bibliothèque ne présente d'intérêt que si elle est indépendante du cas particulier du projet "Inpres-PFM" ... Ainsi :

bien ☺	pas bien ☹
<pre>xxx <b>receiveSize</b>(void * struc, int size) /* couche basse : réutilisable dans une autre application */ xxx <b>receiveSep</b>(char *chaine, char *sep) /* couche basse : réutilisable dans une autre application */ ListeContainers <b>GetOutputReady</b> (...,...) /* couche haute : propre à cette application - utilise l'une des fonctions ci-dessus */</pre>	<pre>xxx receive(ListeContainers *lc, int size) // et pas de xxx GetOutputReady (...,...) /* une seule couche : la fonction receive ne peut être utilisée dans une autre application */</pre>

4.3) Une troisième remarque pleine de bon sens ;-): multiplier les couches exagérément est certainement le meilleur moyen de compliquer le développement et de ralentir l'exécution !

4.4) Enfin, en tenant compte de l'administration du serveur, il serait avisé de faire intervenir dans le code du serveur **la notion d'état** de celui-ci (*certaines commandes n'ont de sens que si elles sont précédées d'une autre*).

5) Il est impérieux de surveiller les écoutes, les connexions et les communications réseaux  
- au moyen des commandes **ping** et surtout **netstat** (savoir expliquer les états TCP);  
- en utilisant un **sniffer** comme Wireshark ou autre encore analysant le Mouvements réseau (attention au localhost qui ne permet pas de sniffer simplement). Cette pratique sera demandée lors des évaluations.

6) Il serait aussi intéressant de prévoir un fichier de configuration lu par le serveur à son démarrage. A l'image des fichiers propriétés de Java, il s'agit d'un simple fichier texte dont chaque ligne comporte une propriété du serveur et la valeur de celle-ci :

serveur_production.conf
<pre>Port_Mouv=70000 Port_Admin=70009 sep-trame=\$ fin-trame=# sep-csv=, pwd-prepose-sud=supersexy pwd- prepose-ouest=maitrevcmw ...</pre>

## Les travaux de l'évaluation 2 : JDBC et client-serveur multithread TCP en Java

### Compétences développées :

- ◆ Mettre en œuvre les techniques de threads et JDBC en Java;
- ◆ Maîtriser l'implémentation du modèle client-serveur sur base des sockets TCP en Java;
- ◆ Mettre en œuvre les techniques de threads et JDBC en Java;

### Dossier attendu :

1. schéma relationnel de BD\_ MOUVEMENTS;
2. diagramme de classes UML des classes de database.facility.
3. définition et implémentation des commandes du protocole TRAMAP;
4. code Java du serveur Serveur\_Mouvements;
5. trames échangées et vues par un sniffer.

## **9. Les accès aux bases de données**

### **9.1 La base de données BD\_ MOUVEMENTS**

Cette base MySQL BD\_ MOUVEMENTS doit contenir toutes les informations utiles concernant la gestion de la plate-forme multimodale. Cependant, ce qui concerne les aspects financiers (paiements des transports, frais de dépôts, frais complémentaires, etc) est du ressort d'une autre base BD\_COMPTA (c'est la volonté du service de disposer d'une base séparée).

Ses tables, si on admet un certain nombre d'approximations, de simplifications et d'omissions sans importance pour le projet tel que défini ici, seront en première analyse :

- ◆ **Parc** : il s'agit ici de l'équivalent du fichier FICH\_PARC évoqué ci-dessus;
- ◆ **Sociétés** : les différentes sociétés qui traitent avec la plate-forme :
  - \* identifiant société;
  - \* nom du contact;
  - \* email du contact;
  - \* numéro de téléphone de ce contact;
  - \* adresse de la société (simple chaîne de caractères).
- ◆ **Containers**: les différents containers qui transitent ou ont transité par le parc :
  - \* identifiant container;
  - \* identifiant société propriétaire;
  - \* nature du contenu;
  - \* capacité
  - \* dangers particuliers.
- ◆ **Transporteurs**: les différents transporteurs (routiers, fluviaux ou ferroviaires) qui ont traité avec la plate-forme :
  - \* identifiant transporteur (immatriculation dans le cas d'un camion);
  - \* identifiant société propriétaire;
  - \* capacité
  - \* caractéristiques techniques (sous forme d'une simple chaîne de caractères).
- ◆ **Mouvements** : tout mouvement d'un container en entrée ou en sortie :
  - \* identifiant mouvement
  - \* identifiant container
  - \* identifiant transporteur entrant
  - \* date arrivée



- \* identifiant transporteur sortant
- \* poids (contenant + contenu);
- \* date départ
- \* destination

♦ **Destinations** : les destinations desservies par la plate-forme :

- \* ville
- \* distance par bateau (depuis la plate-forme)
- \* distance par train
- \* distance par route

On a bien sûr toute liberté pour ajouter des champs supplémentaires, voire d'autres tables ou vues – mais de manière limitée.

## **9.2 Une classe d'accès aux bases de données**

L'accès à la base de données ne devrait pas se faire avec les primitives JDBC utilisées telles quelles, mais plutôt au moyen d'objets métiers encapsulant le travail, de type Java Beans simples, donc sans la notion d'événements ou de factory. On demande donc de construire un groupe de classes (**BeanBDAccessxxx ...**) permettant l'accès (c'est-à-dire à tout le moins la connexion et les sélections de base) le plus simple possible. On souhaite pouvoir accéder, au minimum, à des bases relationnelles de type MySQL et Oracle

Un programme de test préalable serait plus prudent.

## **10. Le serveur Serveur Mouvements (1)**

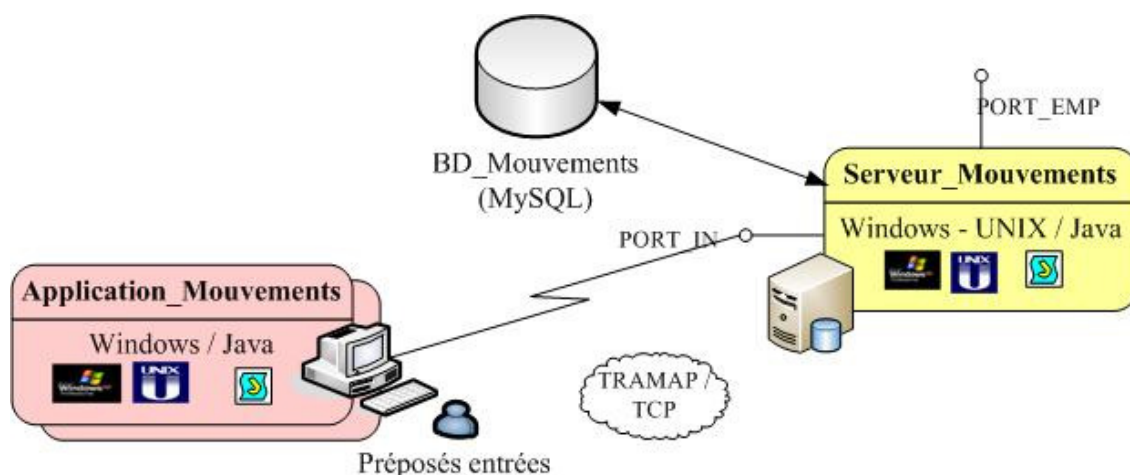
Nous allons ici nous préoccuper de l'implémentation du modèle client-serveur pour le serveur **Serveur\_Mouvements** (client: **Application\_Mouvements**).

### **10.1 Serveur Mouvements et Application Mouvements**

Ce serveur est donc un serveur multithread Java/Windows-Unix (en modèle pool de threads) qui est chargé de gérer le Mouvements des containers : leur arrivée et leur départ, leur stockage, les réservations et prolongations, leur traçage, etc; il utilise la base de données MySQL BD\_MOUVEMENTS.

Il attend ses requêtes sur

- ♦ le port PORT\_IN pour la gestion des arrivées de camions à l'entrée de la plate-forme;
- ♦ le port PORT\_EMP pour les requêtes concernant les emplacements, requêtes provenant du Serveur\_Containers (voir évaluation 4).



## 10.2 La gestion des arrivées

Le serveur utilise le protocole applicatif (basé TCP) **TRAMAP** (Mouvements MAnagement Protocol), dont les commandes sont :

protocole <b>TRAMAP</b> - PORT_IN		
application cliente : <b>Application_Mouvements</b>		
commande	sémantique	réponse éventuelle
LOGIN	Un préposé veut se connecter <i>paramètres</i> : nom et mot de passe	oui + attente d'une autre commande ou non + cause
INPUT_LORRY	Arrivée d'un camion apportant un container pour lequel une réservation a été faite <i>paramètres</i> : numéro de réservation + identifiant du container qu'il transporte	oui + emplacement (x,y) qui a été attribué à ces containers ou non + cause (par exemple, un identifiant de container ne correspond pas)
INPUT_LORRY_WITHOUT_RESERVATION	Arrivée d'un camion apportant un container SANS réservation <i>paramètres</i> : numéro d'immatriculation du camion + identifiant du container qu'il transporte	oui + emplacement (x,y) qui a été attribué à ces containers ou non (plus de place)
LIST_OPERATIONS	liste des mouvements dans un certain intervalle de dates pour une société donnée ou une destination donnée <i>paramètres</i> : les dates, le nom de la société ou la destination	oui + la liste ou non (critère de recherche inconnu)
LOGOUT	Un préposé veut se déconnecter <i>paramètres</i> : nom et mot de passe	oui + session fermée ou non + cause

## Les travaux de l'évaluation 3 : caddie virtuel en Java

### Compétences développées :

- ♦ Maîtriser les techniques classiques du développement Web en Java.

### Dossier attendu :

1. schéma des deux applications Web en termes de servlets, JSPs, pages HTML (formalisme libre).

## **11. L'application Web Applic Reservation**

L'objectif est de permettre aux entreprises extérieures de réserver des emplacements et des transferts pour leurs containers. On utilisera ici **HTTP** avec la technologie des servlets Java et des Java Server Pages.

Il est simplement demandé de réaliser ici le login et la réservation d'un container, ceci afin de se familiariser très vite avec les concepts de base de la programmation Web en Java. Une autre application Web, plus aboutie, sera proposée au point suivant.

Une page HTML basique propose à l'utilisateur de se connecter à l'application Web en permettant

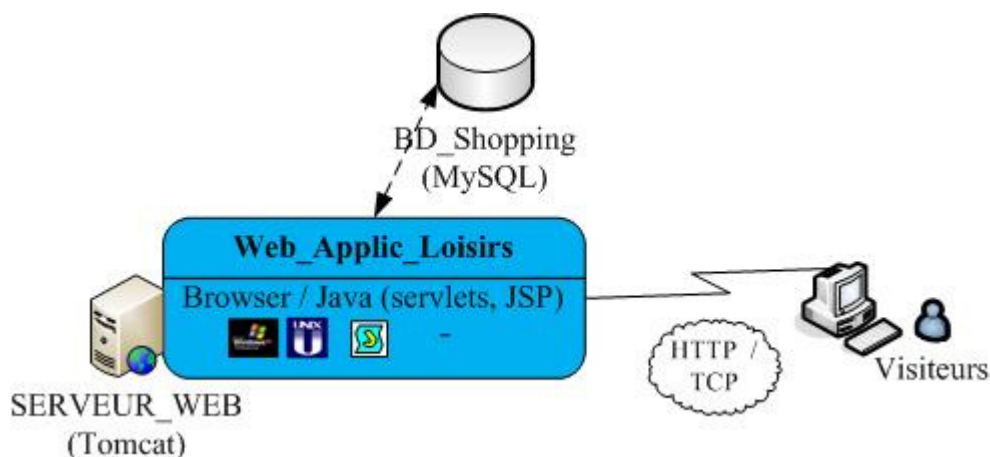
- a) d'y entrer son identifiant et son mot de passe (une case à cocher spécifie si il s'agit d'un nouveau client) qui sera vérifié/enregistré côté serveur par une servlet;
- b) de recevoir de cette servlet une page HTML dynamique comportant un formulaire de demande de réservation d'emplacement pour une destination donnée : cette destination est à choisir dans une liste fournie par la servlet à partir de BD\_MOUVEMENTS (au moyen du bean BeanBDAccess.xxx); le choix effectué est envoyé à la servlet qui recherche un emplacement libre dans BD\_MOUVEMENTS (*en fait, elle devrait s'adresser à Serveur\_Mouvements qui gère ces réservations; mais, pour simplifier, la servlet attaquera directement BD\_MOUVEMENTS*);
- c) d'obtenir enfin de la servlet une JSP donnant le résultat de la requête : soit un numéro de réservation (accompagné du numéro de l'emplacement réservé - pour information car cet emplacement pourrait être modifié ultérieurement pour des raisons de service), soit un refus poli.



## 12. L'application Web Web\_Applic Loisirs

L'application Web Web\_Applic Loisirs, basée sur des technologies servlets/JSP, qui permet aux particuliers de réserver des visites au parc de loisirs verts et/ou à la réserve naturelle, ou simplement d'acheter des guides natures et autres produits en relation (peluches d'animaux, accessoires de bureau "nature", tenues "Tarzan" et "Jane", etc).

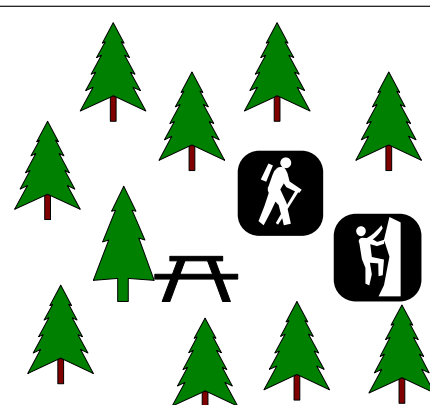
Les pages HTML et les servlets, les JSPs, bref les applications Web, seront déployées sur un serveur Tomcat (de préférence non local) tournant sur un PC. Les informations nécessaires à la gestion des clients et de leurs caddies seront mémorisées dans une petite base de données BD\_Shopping (à construire selon les besoins de l'application).



### 12.1 Le login

La procédure de login est gérée par une servlet spécifique `LoginServlet` et utilise un interface constitué de pages HTML:

- ♦ une page HTML demandant l'introduction d'un identifiant et d'un mot de passe, donc selon le canevas suivant :

Bienvenue sur PFM Loisirs ! Le site des loisirs verts près de chez vous ☺	
Votre identifiant :	<input type="text"/>
Votre mot de passe :	<input type="password"/>
<input type="button" value="Entrer sur le site"/>	
<input type="button" value="Je n'ai pas de mot de passe"/>	
	


"Entrer sur le site" correspond à un client qui possède déjà un mot de passe (donc qui s'est déjà identifié par le passé), "Je n'ai pas de mot de passe" à un nouveau client qui n'en possède



pas encore (ou à un client qui l'a oublié). Dans ce dernier cas, il peut créer un mot de passe au moyen d'une autre page :

♦ cette deuxième page se présente sous la forme du canevas suivant (elle récupère l'identifiant obtenu dans la première page) pour un client qui a entré comme identifiant "Dugenou" :

Bienvenue sur PFM Loisirs ! Le site des loisirs verts près de chez vous ☺	
Votre identifiant :	Dugenou
Choisissez un mot de passe :	<input type="password"/>
Confirmer votre mot de passe :	<input type="password"/>
<input type="button" value="Confirmer"/>	<input type="button" value="Abandonner"/>
<input type="button" value="Aide"/>	





L'appui sur le bouton "Confirmer" provoquant la recopie du mot de passe dans la zone "Votre mot de passe" de la première page de login.

L'appui sur le bouton "Entrer sur le site" de cette première page visera une page HTML dynamique (voir ci-dessous - une page HTML statique pourrait évidemment être accédée sans login préalable).

## 12.2 Les achats



En cas de succès du login du client, la page HTML obtenue propose trois services selon le canevas suivant :

PFM Loisirs ! Le site des loisirs verts près de chez vous ☺ – Session Dugenou	
<input checked="" type="radio"/> visites au parc de loisirs verts et/ou à la réserve naturelle	
<input checked="" type="radio"/> achats de guides et objets « nature »	
<input type="button" value="Continuer"/>	<input type="button" value="Abandonner"/>

Techniquement, ce site est construit selon le modèle MVC pour obtenir un billet d'entrée au parc et/ou la réserve naturelle (attention : pour cette dernière, le nombre de visiteurs est limité à 20 par jour) et pour le caddie virtuel qui permet à l'utilisateur de réaliser

des achats des produits "nature" se trouvant dans le stock: ceci consiste à se promener dans les pages catalogues du site de Inpres-PFM pour y choisir au fur et à mesure des articles.

Finalement, on en arrive à réaliser les opérations de paiement.

PFM Loisirs ! Le site des loisirs verts près de chez vous ☺ – Session Dugenou	
<input checked="" type="radio"/> fin de session et paiement	 
<input type="button" value="Continuer"/>	<input type="button" value="Abandonner"/>

Plus précisément pour le caddie virtuel, on utilisera des Java Server Pages :

- ◆ **JSPInit** : Bonjour et initialisation du caddie
- ◆ **JSPCaddie** : Choix et ajouts au caddie
- ◆ **JSPPay** : Confirmation des achats et envoi au paiement par carte de crédit (simple encodage, aucune sécurité n'est assurée à ce stade).

En ce qui concerne **la politique de gestion des caddies**, les principes suivants seront d'application :

- ◆ si un client demande une quantité d'articles supérieure à celle en stock, il peut réserver le maximum d'articles disponibles mais seulement si ceci agrée le client;
- ◆ tout ce qui est placé dans un caddie est considéré comme une promesse ferme : autrement dit, pour peu que le client termine son marché **dans un délai raisonnable, il est assuré que ce qu'il a réservé ne peut être pris par un autre client;**
- ◆ si un client tarde plus d'une demi-heure (par exemple) ou encore si il ferme son navigateur sans avoir conclu, ce qui se trouvait dans son caddie est remis sur le marché;
- ◆ tout achat validé doit bien sûr être mémorisé dans la base de données (pratique normale du commerce).

## Les travaux de l'évaluation 4 : client-serveur sécurisé en Java et caddie virtuel en Java

### Compétences développées :

- ◆ Maîtriser les concepts et l'implémentation des techniques cryptographiques classiques en Java;
- ◆ Maîtriser les techniques classiques du développement Web en Java.

### Dossier attendu :

2. explication du contenu des différents keystores;
3. diagramme du handshake pour l'échange des clés publiques;
4. schéma de l'application Web en termes de servlets, JSPs, pages HTML (formalisme libre).

## **13. Le serveur Serveur Compta**

### **13.1 La base de données BD Compta**

Deux bases de données interviennent donc ici :

- ◆ BD\_Mouvements : déjà décrite;
- ◆ BD\_Compta : doit contenir les informations utiles concernant tout ce qui touche les mouvements financiers de la plate-forme (correspondant à l'utilisation des containers) et les salaires du personnel (que nous ne traiterons pas ici).

Pour cette dernière base, nous avons besoin seulement, pour l'instant, des tables :

- ◆ **Personnel** : les membres du personnel de la plate-forme :

- \* identifiant matricule
- \* nom
- \* prénom
- \* login
- \* mot de passe
- \* adresse e-mail interne
- \* fonction (manutentionnaire, chef d'équipe, préposé(e), chef de poste, comptable, chef-comptable, directeur(ice), ...)

- ◆ **Items Facture** : une ligne de facture correspondant à un mouvement

- \* identifiant item (compteur)
- \* identifiant facture;
- \* identifiant mouvement;
- \* identifiant container;
- \* destination
- \* destination;
- \* prix (hors TVA);

◆

- ◆ **Factures** : pour une société donnée, on facture tous les mouvements du mois écoulé; outre son numéro et ses Item\_Factures, une facture comportera

- \* l'identifiant de la société;
- \* le mois et l'année considérés;
- \* le montant total hors TVA;

- \* le montant total à payer (donc TVA comprise);
- \* un flag "facture validée"
- \* l'identifiant du comptable valideur
- \* un flag "facture envoyée"
- \* le moyen d'envoi de la facture (mail ou papier)
- \* un flag "facture payée"

♦ **Tarifs** : prix à payer pour un emplacement et sa manipulation (le prix n'est pas le même selon qu'interviennent un camion et/ou un bateau et/ou un train - on peut se référer à <http://www.portdeliege.be/fr/tarification>) :

- \* identifiant prix
- \* type (occupation emplacement ou manipulation)
- \* matériel utilisé (pont roulant pour occupation, camion, bateau ou train pour manipulation)
- \* flag produit pétrolier
- \* date dernière mise à jour.

On a bien sûr toute liberté pour ajouter des champs supplémentaires, voire d'autres tables ou vues – mais de manière limitée.

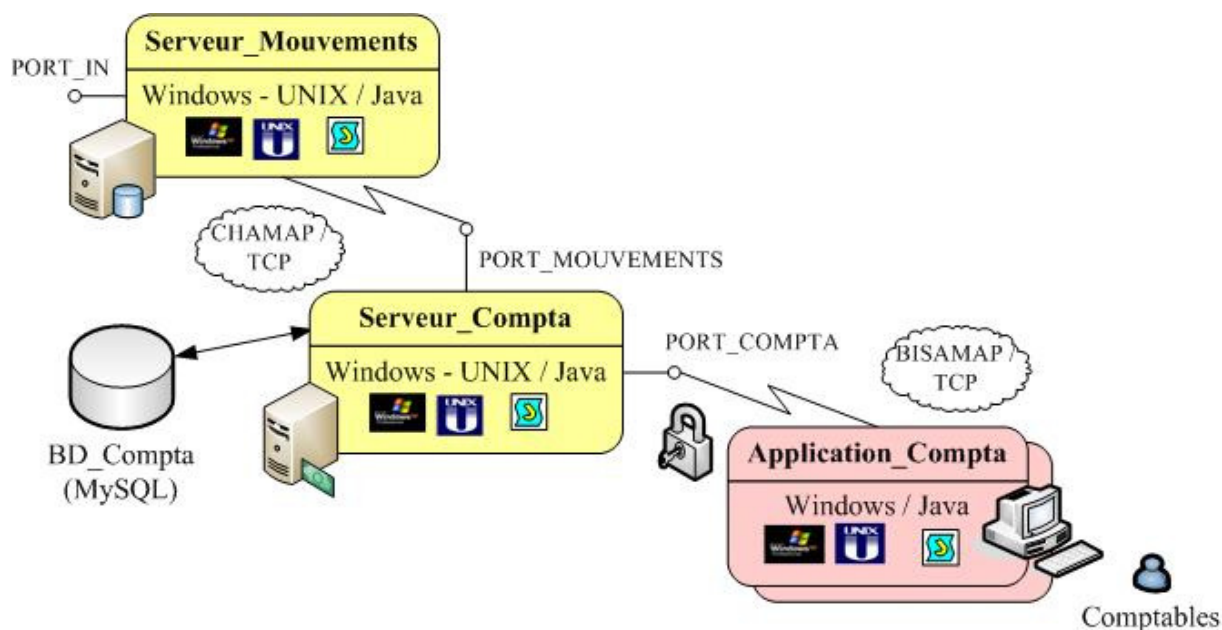
### 13.2 Présentation du serveur

Il gère donc tout ce qui touche aux aspects financiers : paiements des locations et des transferts, paie du personnel, factures de fonctionnement, etc. Ce serveur multithread **Serveur\_Compta** est Java/Windows-Unix (en modèle pool de threads). Il attend ses requêtes sur

♦ le port PORT\_MOUVEMENTS pour les requêtes émanant du **Serveur\_Mouvements**, requêtes portant essentiellement sur la génération des factures;

♦ le port PORT\_COMPTA pour les requêtes provenant des applications clientes **Application\_Compta** (utilisée par les comptables), requêtes concernant les paiements des factures de stockages ainsi que les mouvements et transports par les sociétés utilisatrices.

Comme il s'agit de traiter des informations que l'entreprise entend garder confidentielles et sûres, il faut envisager que les communications réseaux soient protégées au moyen des outils de cryptographie (clé secrète, clés publique et privée, message digest, signature digitale, HMAC).



### 13.3 Les factures de stockage et de transport

Le serveur utilise sur le port PORT\_MOUVEMENTS le **protocole applicatif** (basé TCP) **CHAMAP** (**CH**arge **MA**nagement **P**rotocol), dont les commandes sont simplement

protocole <b>CHAMAP</b> - PORT_MOUVEMENTS		
application cliente : <b>Serveur_Mouvements</b>		
commande	sémantique	réponse éventuelle
LOGIN_ TRAF	le serveur Serveur_Mouvements veut se connecter (il dispose d'un identifiant et d'un mot de passe convenus) - la connexion sera maintenue <i>paramètres</i> : nom et <b>digest "salé"</b> du mot de passe	oui ou non
MAKE_BILL	provoque la génération des factures pour les containers embarqués à partir des informations de mouvements <i>paramètres</i> : identifiant bateau ou train + liste des identifiants des containers effectivement embarqués	au choix

### 13.4 Les activités comptables : les factures

Le serveur utilise sur le port PORT\_COMPTA le **protocole applicatif** (basé TCP) **BISAMAP** (**BI**lls and **SA**lary **MA**nagement **P**rotocol), dont les commandes utilisant des techniques cryptologiques, sont

protocole <b>BISAMAP</b> - PORT_MOUVEMENTS		
application cliente : <b>Application_Compta</b>		
commande	sémantique	réponse éventuelle
LOGIN	un comptable veut se connecter <i>paramètres</i> : nom et <b>digest "salé"</b> du mot de passe	oui + lancement de la procédure de handshake pour le partage des deux clés symétriques ou non
GET_NEXT_BILL	pour obtenir la facture la plus ancienne non encore validée <i>paramètres</i> : -	oui + la facture <b>chiffrée symétriquement</b> + attente de la commande suivante ou non (pas de facture)
VALIDATE_BILL	valider cette facture si elle est correctement libellée ou invalider <i>paramètres</i> : numéro de facture, <b>signature</b> du comptable	oui (signature vérifiée) + confirmation ou non (signature non vérifiée)
LIST_BILLS	pour obtenir la liste de toutes les factures (payées ou non) d'une société donnée pour un intervalle de temps donné <i>paramètres</i> : identifiant société, dates de l'intervalle, <b>signature</b> du comptable	oui (signature vérifiée) + liste <b>chiffrée symétriquement</b> ou non

SEND_BILLS	ordre d'envoyer les factures validées par le comptable sauf peut-être certaines qu'il précise <i>paramètres</i> : liste des factures à ne pas envoyer (éventuellement, liste vide), <b>signature</b> du comptable	oui (signature vérifiée) ou non (certaines factures "à éviter" n'existent pas)
REC_PAY	REceiving PAYment: enregistrement du paiement d'une facture <i>paramètres</i> : numéro de la facture, montant, informations bancaires + <b>chiffré symétriquement</b> + HMAC du comptable	oui (HMAC vérifié) + confirmation ou non + pourquoi (ex: montant payé différent du montant attendu)
LIST_WAITING	List of WAITING payments : liste des factures non encore payées : toutes ou celles d'une société donnée ou celles qui ont été émises depuis plus d'un mois <i>paramètres</i> : indications sur la nature de la liste, <b>signature</b> du comptable	la liste ou non + pourquoi

Les mécanismes de cryptographie **asymétrique** utilisent des clés publiques et privées. On peut, dans un premier temps, se contenter de clés sérialisées dans des fichiers. Mais, dans un deuxième temps, les clés asymétriques et les certificats doivent se trouver dans des **keystores**.

Dans un premier temps, on peut considérer que les clés secrètes **symétriques** sont sérialisées dans des fichiers. Mais, dans un deuxième temps, ces clés secrètes doivent être échangées au moyen d'un chiffrement asymétrique : il y a donc alors une phase préalable de **handshake**.

Enfin, la clé symétrique utilisée par un comptable pour s'authentifier (HMAC) n'est donc pas la même que celle qui lui permet de chiffrer/déchiffrer : tout comptable possède donc deux clés symétriques à usage différent.

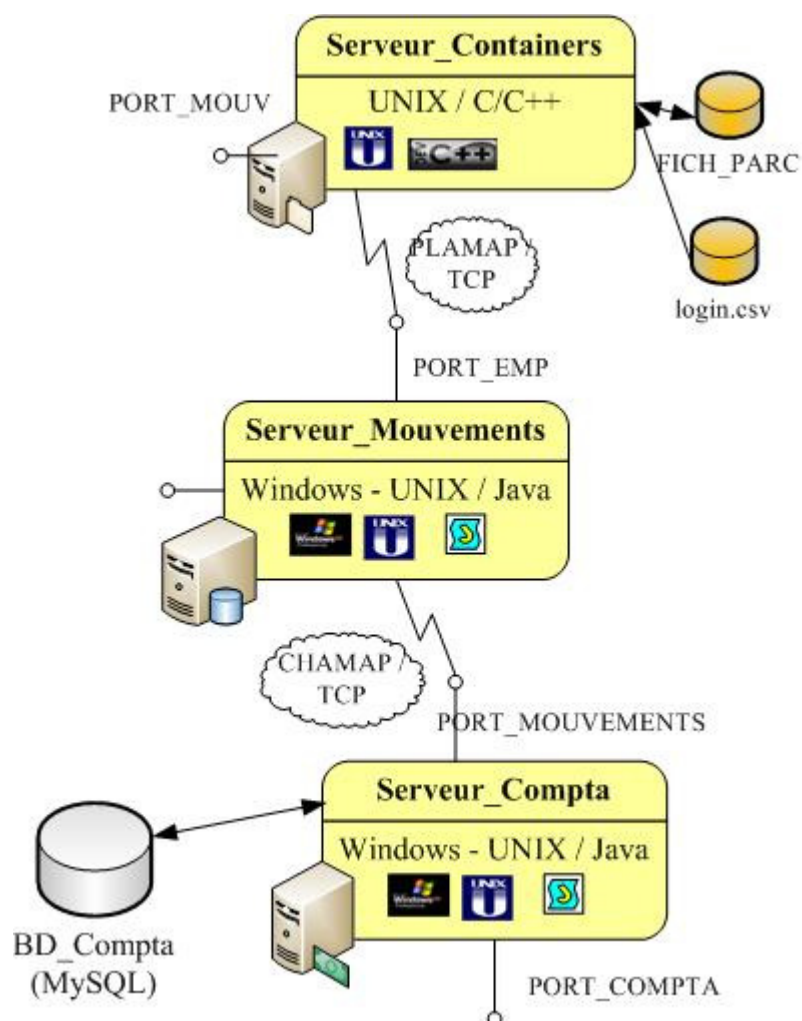
#### **14. Le serveur Serveur Mouvements (2) : la gestion des emplacements**

Le serveur utilise le **protocole applicatif** (basé TCP) **PLAMAP** (PLAce MAagement Protocol), dont les commandes sont :

protocole <b>PLAMAP</b> - PORT_EMP		
application cliente : <b>Serveur_Containers</b>		
commande	sémantique	réponse éventuelle
LOGIN_CONT	Serveur_Containers veut se connecter (il dispose d'un identifiant et d'un mot de passe convenus) - la connexion sera maintenue <i>paramètres</i> : nom et mot de passe	oui ou non
GET_XY	GET emplacement (x,y) : pour obtenir l'emplacement réservé dans le parc pour le camion qui va être déchargé <i>paramètres</i> : identifiant société propriétaire + numéro d'immatriculation du camion + identifiant du container qu'il transporte + destination	oui + numéro de réservation + emplacement (x,y) qui a été attribué à ces containers ou non + pourquoi (un des renseignements envoyés ne correspond pas)



SEND_WEIGHT	Pour envoyer le poids réel du container <i>paramètres</i> : identifiant du container + emplacement + poids	oui ou non (container inconnu)
GET_LIST	Pour obtenir la liste des emplacements occupés par des containers pour une destination donnée <i>paramètres</i> : identifiant bateau ou train, destination, nombre maximum de containers transportables	oui + la liste des emplacements, par ordre d'arrivée ou non (pas de containers ou destination inconnue)
SIGNAL_DEP	SIGNAL DEParture : un bateau ou un train a terminé ses opérations de chargement et quitte le site <i>paramètres</i> : identifiant bateau ou train + liste des identifiants des containers effectivement embarqués	oui + (voir plus haut), commande <i>MAKE_BILL</i> envoyée à <i>Serveur_Compta</i> ou non (transporteur inconnu)



## 15. Inpres-PFM-Chat

### 15.1 Fonctionnalités demandées

Les ouvriers, employés et cadres de l'entreprise (terme générique : les "agents") peuvent discuter par chat pour s'informer sur le temps qu'il fait, la robe de la directrice des ventes, la cravate hideuse du directeur général, le physique de la nouvelle secrétaire, la chute de cheveux du responsable des achats ou autres informations (f)utiles. C'est le " Inpres-PFM-Chat".

On est admis dans ce chat sur base de ses informations d'identification sur le serveur dont on dépend, donc son nom et mot de passe.

### 15.2 Client UDP en Java

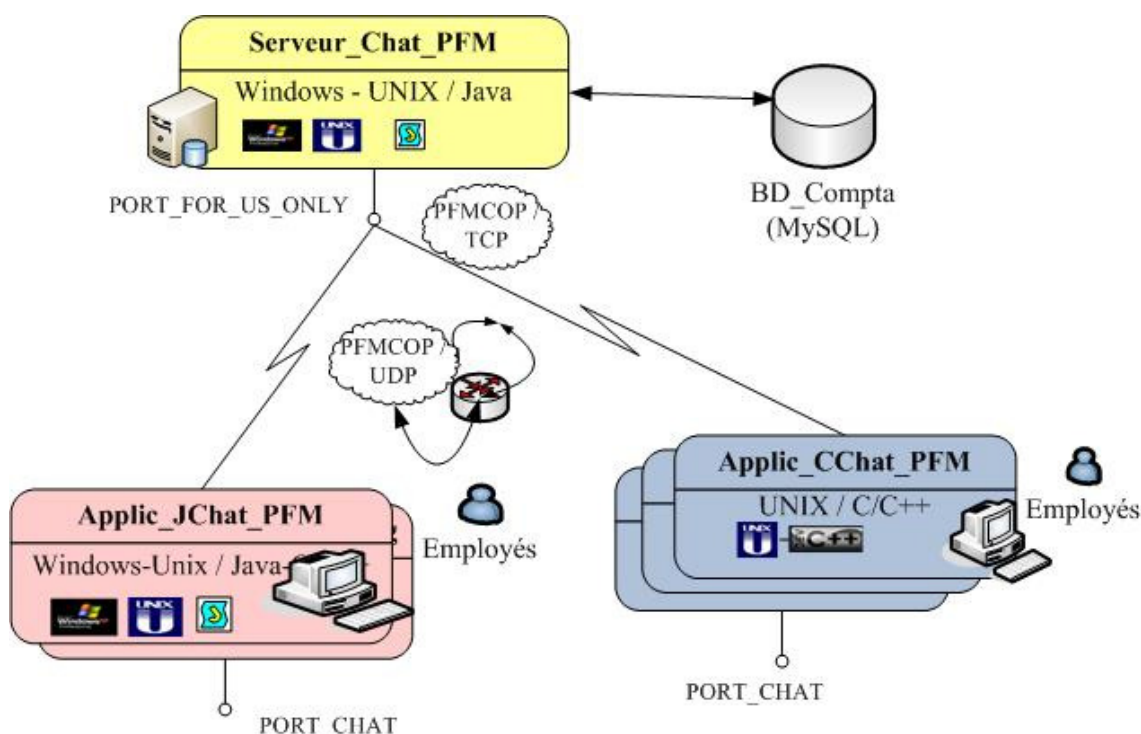
Au moyen d'une application **Application\_JChat\_PFM** écrite en Java, on se joint au groupe en UDP sur une adresse de classe D précise et un port PORT\_CHAT précis qui aura été obtenu en s'adressant sur base TCP à un serveur **Serveur\_Chat\_PFM** qui vérifie l'appartenance à la Inpres-PFM dans la base donnée BD\_COMPTA contenant les informations nécessaires à l'identification.

Les agents utilisent pour cela le protocole **PFMCOP (PFM Community cOnversation Protocol)**. Basé principalement UDP, ce protocole applicatif utilise donc cependant au préalable une connexion TCP à **Serveur\_Chat\_PFM** écrit en Java sous Windows; celui-ci n'attend sur le port PORT\_FOR\_US\_ONLY qu'une seule commande permettant de fournir le nom et le mot de passe de l'agent :

protocole <b>PFMCOP (partie TCP)</b> - PORT_FOR_US_ONLY		
application cliente : <b>Application_JChat_PFM</b>		
commande	sémantique	réponse éventuelle
LOGIN_GROUP	un agent veut se joindre au groupe <i>paramètres</i> : nom et <b>digest "salé"</b> du mot de passe	oui + envoi de l'adresse et du port PORT_CHAT à utiliser ce jour; ou non

En cas de succès, l'agent pourra alors réellement participer au chat :

protocole <b>PFMCOP (partie UDP)</b> - PORT_CHAT		
application cliente : <b>Application_JChat_PFM</b>		
commande	sémantique	réponse éventuelle
POST_QUESTION	Pose question : un agent pose une question et espère des réponses <i>paramètres</i> : un tag d'identification de question à utiliser dans la réponse (généralisé), la question sous forme de chaîne de caractères accompagnée d'un <b>digest</b> pour contrôler l'intégrité	une réponse à la question précédée du tag
ANSWER_QUESTION	Réponse à une question après vérification de son digest. <i>paramètres</i> : le tag d'identification de question et le texte de la réponse	une réponse à la question précédée du tag
POST_EVENT	Un agent signale un fait, donne une information mais n'attend pas de réponse (un deuxième type de tag est cependant généré pour identifier l'événement)	-



Il s'agira évidemment de mettre d'abord en place le **Serveur\_Chat\_PFM**, qui est un serveur Java qui tourne sur un PC et qui se révèle, du point de vue TCP, des plus simples puisque monothread et mono commande. A noter tout de même qu'il appartient aussi au groupe multicast UDP. Le multicast sera ensuite implémenté, tout d'abord dans le sous-réseau local.

### 15.3 Client UDP en C/C++

On élargira ensuite le chat en ajoutant une application **Application\_CChat\_PFM** écrite en C/C++ (car les ateliers utilisent plutôt des applications écrites dans ces langages).

## 16. L'administration basique du serveur **Serveur Containers**

Le serveur **Serveur Containers** attend sur le **PORT\_ADMIN** les requêtes d'une application **Application\_Admin** pour permettre aux responsables informatiques d'administrer ce serveur. Un client de ce type peut donc administrer le serveur à distance du point de vue technique (arrêt, reprise, liste des machines du réseau). L'application **Application\_Admin** est programmée en Java et fonctionne sur un PC Windows ou sur une machine UNIX (machine Sunray); elle utilise le protocole **CSA (Container Server Administration)**. Ce protocole applicatif, (basé TCP), a pour commandes :

protocole <b>CSA</b> - <b>PORT_ADMIN</b>		
application cliente : <b>Application_Admin</b>		
Commande	sémantique de la requête	réponse éventuelle
LOGINA	un administrateur autorisé se connecte <i>paramètres</i> : nom, password	oui ou non (par exemple : un autre administrateur est déjà connecté et il ne peut y en avoir qu'un seul à la fois)

LCLIENTS	List CLIENTS : liste des préposés connectés paramètres : -	pour chaque client : adresse IP
STOP	STOP Server: on réalise un shutdown du serveur en prévenant les clients de l'imminence de l'arrêt <i>paramètres</i> : nombre de secondes avant arrêt	oui ou erreur

Le client administrateur se connecte sur **le port d'administration** PORT\_ADMIN et se fait reconnaître comme administrateur en introduisant un mot de passe propre à l'administration. Pour que les clients connectés soient avisés, *de manière asynchrone*, d'un arrêt temporaire ou d'un shutdown du serveur (dans ce dernier cas, afin de se terminer en douceur), il convient de choisir une stratégie :

- ou chaque client normal s'est connecté sur un deuxième port (PORT\_URGENCE) et attend des messages urgent par cette voie;
- ou le serveur se connecte sur chaque client en cas de nécessité (le client aura envoyé son port d'écoute lors de sa connexion sur PORT\_ADMIN).

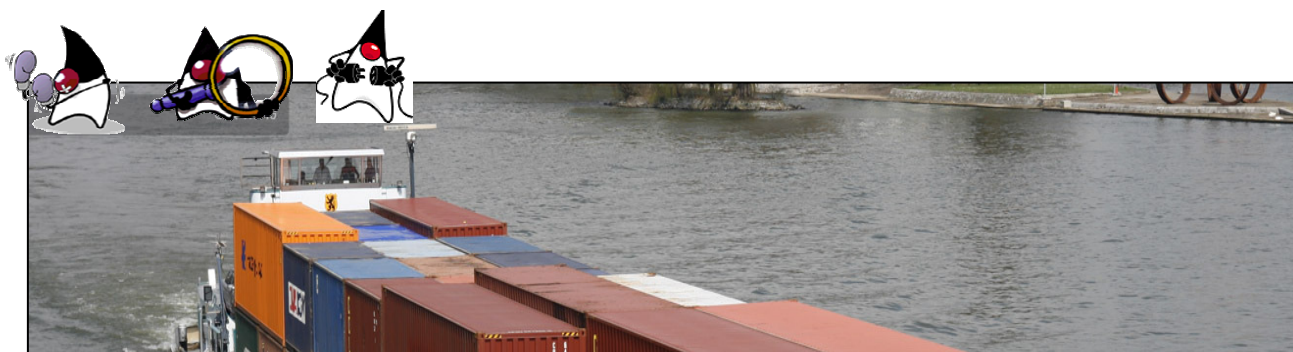
### En guise de conclusion

Les principaux développements évoqués dans ce dossier de laboratoire ont été définis avec le plus de précision possible.

Certains points ont cependant été laissés suffisamment vagues pour vous permettre d'envisager différents scénarii pour satisfaire à la demande. De plus, certaines pistes sont à peine entr'ouvertes - à vous de voir, *si vous en avez le temps*, ce que vous pouvez ajouter à vos développements pour leur apporter un plus valorisant. Comme toujours, **prudence** : l'avis de l'un des professeurs concernés est sans doute (un peu-beaucoup-très fort) utile ;-).

Soyez créatifs et imaginatifs ... mais restez rationnels et raisonnables ...

s: CV, CC & SB



*Infos de dernière minute ? Voir l'Ecole virtuelle*