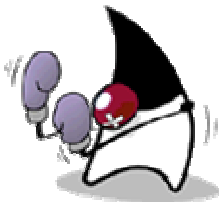
Laboratoire de Compléments programmation réseau (Protocoles, Sécurité logicielle, Mobiles, XML) : projet "ScienceAirport"- suite

3ème Informatique et système – option Réseaux-télécommunications 2022-2023

Projet *InPrESEnjoyYourHolidays* - la suite

Claude Vilvens et Christophe Charlet



1. Le contexte

L'Unité d'Enseignement "**Programmation réseaux, web et mobiles**" (10 ECTS - 135h) se structure en Informatique de gestion en trois Activités d'apprentissage de la manière suivante :

- AA: Réseaux et technologies Internet (60h - 45%)
- AA: Programmation.Net (30h - 22%)
- AA: Technologie de l'e-commerce et mobiles (45h - 33%)

Le contexte de ce laboratoire de "Compléments Programmation Réseaux" est le même que celui du laboratoire de "Réseaux et technologies Internet", à savoir celui de "InpresEnjoyYourHolidays" qui vise à la gestion d'une infrastructure touristique de villages de vacances.

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 assuré 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 "Compléments Programmation Réseaux", 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 "Compléments Programmation Réseaux" est réalisée de la manière suivante :

- examen de théorie: un examen **oral** en janvier 2023 (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 en évaluation continue: une évaluation ("évaluation 1" ci-dessous) cotée sur 20 qui constitue la note d'évaluation continue (non remédiable en 2ème session);
- examen de laboratoire: un examen oral en janvier 2023 consistant en la présentation de la 2ème partie du laboratoire ("évaluation 2" ci-dessous) et coté sur 20;
- note finale : **moyenne géométrique de la note de l'examen de théorie (poids de 50%), de la note d'évaluation continue (poids de 15%) et de la note de l'examen de laboratoire (poids de 35%)**.

Dans ces conditions, ***il est clair qu'une note beaucoup trop basse parmi les trois ne peut que conduire à l'échec de l'AA considérée***.

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: keystore SSL, régression multiple et tests, etc).

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

- la note de théorie;
- la note de laboratoire de l'évaluation 2 (examen de laboratoire).

Les évaluations de théorie et du laboratoire d'examen 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). La pondération est de 50% et 50%.

La note de laboratoire de l'évaluation 1 n'est pas remédiable et n'intervient plus dans le calcul de la note finale (principe général de l'évaluation continue).

Le laboratoire de "Compléments Programmation Réseaux" comportera donc deux évaluations. La première (SNMP, messagerie électronique) sera **évaluée** par l'un des professeurs du laboratoire **à partir du 28 novembre 2022** (avec rentrée d'un dossier papier tel que décrit dans l'énoncé). La deuxième (Android, SSL, XML …) sera évaluée lors de l'examen de laboratoire en **janvier 2023** (un dossier papier ne sera plus nécessaire).

3. Agenda des évaluations

Pour chaque évaluation, le délai est à *respecter* impérativement.

Evaluation	**Evaluation continue : semaine d'évaluation**	**Remédiable en 2ème session**
Evaluation 1 :	28/11/2022-2/12/2022	non
Examen de laboratoire:	Date de l'examen de laboratoire de janvier 2023	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 …

Les travaux de l'évaluation 1: SNMP et messagerie électronique

Compétences développées :
- Savoir utiliser SNMP v1;
- Maîtriser les concepts et techniques de la messagerie électronique;

Dossier attendu :
1. schéma des classes de SNMP4j utilisées;

1. SNMP

1.1 Un petit manager SNMP en Java

Il va de soi que la brillante infrastructure informatique d'Inpres-EnjoyYourHolidays doit s'appuyer sur un réseau et des machines (serveurs et clients) en parfait état de marche.

On a donc mis en place un serveur **Serveur_Control**, de type Java/Windows-Unix, qui fournit les moyens logiciels permettant de gérer valablement le réseau de l'entreprise.

Il est donc muni d'une composante manager SNMP chargée de veiller sur les machines de type PC, comme notamment les machines hébergeant Serveur_Reservations, Serveur_Paiements, Serveur_Activites et Serveur_Materiel. Cette fonctionnalité permet donc simplement au manager SNMP d'envoyer des commandes SNMP et de recueillir des réponses SNMP de manière synchrone ou asynchrone (selon la configuration du serveur - voir fichier de configuration).

Il sera ainsi possible de vérifier que les machines visées (qui sont donc les agents SNMP) sont présentes, d'obtenir divers renseignements comme le nom du contact, le type de machine, l'adresse IP des interfaces, le masque de sous-réseau, etc et de modifier ceux-ci si possible. Toutes les machines concernées font partie de la même communauté SNMP (ici, le nom de votre groupe suivi des initiales des membres du groupe – ex: Charlet, Vilvens: groupe 2921 → communauté = "2921cccv").

On utilisera la librairie **SNMP4J** et MibBowser pour vérifier les résultats de l'application.

1.2 Analyse du trafic réseau et essai de manager SNMP en C

On demande d'analyser avec un sniffer le trafic réseaux pour une requête GET simple et sa réponse.

On demande ensuite de tenter de réaliser la même requête-réponse en C.

2. SMTP/POP3 en Java

2.1 Un mail user agent en Java

Il s'agit tout d'abord de développer une petite application **Application_Mail_Java** qui est un mail user agent, c'est-à-dire permettant de traiter le courrier électronique relatif aux informations échangées par les employés de la société. Plus précisément, elle comporte une fonctionnalité GUI **Java** de gestion classique d'e-mail (type "**mini-outlook**"), permettant d'envoyer et de recevoir un mail
- soit simple de type texte;
- soit composite avec des pièces attachées qui sont des **images** gif ou jpg, ou un **digest** de contrôle d'intégrité ou encore un objet sérialisé (instance de la petite classe **PieceAttachee** qui se limite à encapsuler le type d'information avec les éléments de son contenu.

L'utilisateur devrait idéalement être prévenu dans un délai de 5 minutes de l'arrivée d'un nouveau message. A cet effet, l'application utilisera un thread de polling qui "interrogera" périodiquement la boîte aux lettres.

De plus, une fonctionnalité supplémentaire permettra de "tracer" un message reçu, c'est-à-dire de lister les agents mails (les MTAs) par qui ce message est passé.

2.2 Un second mail user agent en C

Les employés travaillant sur le parc souhaitent utiliser une petite application C **Application_Mail_C** leur permettant d'envoyer des mails limités au format RFC 822.

2.3 Mail de confirmation

L'application **Web_Reservations** envoie un mail de confirmation avec une facture en pièce attachée lorsqu'une réservation est terminée.

Les travaux de l'évaluation 2: Android, SSL et XML

Compétences développées :

- Maîtriser les techniques de base du développement Android.
- Savoir utiliser SSL;
- Savoir utiliser un fichier XML comme fichier de script.

Dossier attendu :

1. schéma général de l'application Android en termes d'activités, d'intents, d'events, de fragments (formalisme libre);
2. la description du fichier XML de configuration.

3. L'application mobile Android pour les réservations

Dans le contexte d' Inpres-EnjoyYourHolidays, l'idée est d'élargir le plus possible les parts de marché d'IEYH, et donc de permettre au grand public d'effectuer des réservations pour les motels et les villages au moyen de son smartphone.

Le serveur **Serveur_Reservations** (évaluation 3 de l'énoncé de "Réseaux et technologies Internet") doit donc aussi satisfaire ces requêtes en considérant que l'application **Mobile_Applic_Reservations** tournant sur le smartphone est capable d'utiliser le protocole ROMP.

Cependant, par mesure de sécurité, cette possibilité n'est offerte qu'aux clients qui sont déjà connus de la société, autrement dit à des voyageurs qui ont déjà effectué une réservation et qui l'ont payée) et qui ont reçu à cette occasion un mot de passe spécial qui leur est propre. Ils devront donc produire ce mot de passe pour pouvoir effectuer une réservation par smartphone.

Techniquement parlant, cette application utilise une architecture basée sur les actvity et les intents.

Vu le contexte, l'internationalisation de l'application est nécessaire.

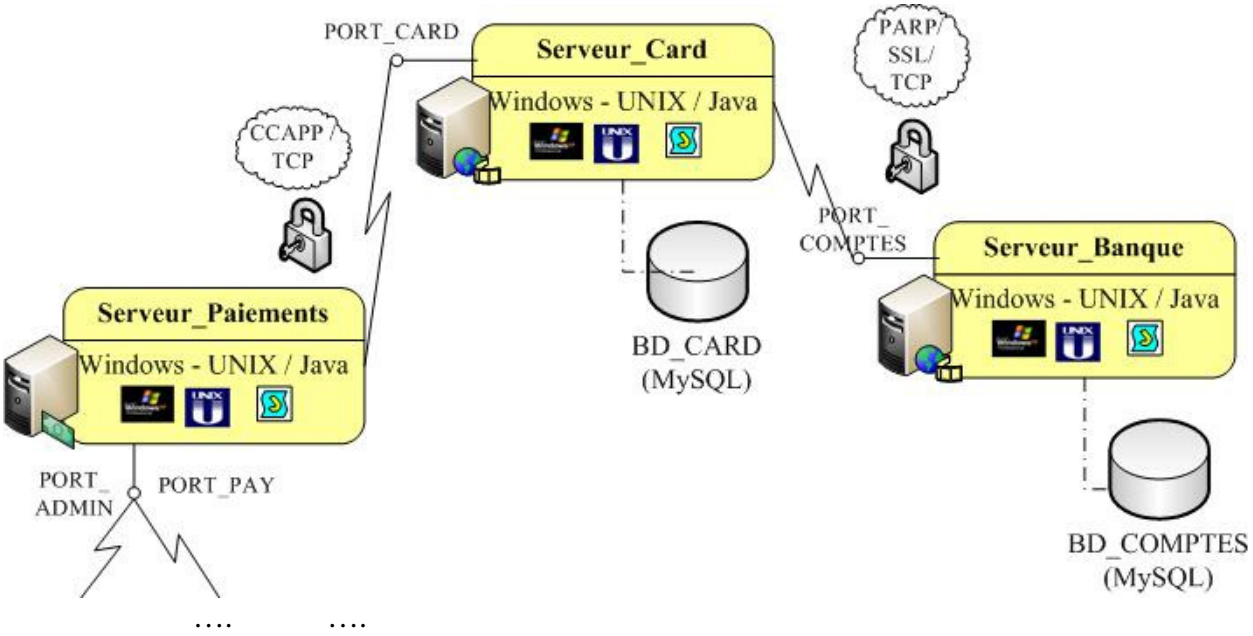
4. Les communications sécurisées par SSL

4.1 La sécurisation des paiements

On se souviendra (peut-être) que, dans la structure de IEHY, il existe un serveur **Serveur_Card**, utilisé par **Serveur_Paiements** et répondant au seul type de requête d'une vérification/débit (protocole **CCAPP** [**C**heck **CA**rd and **P**ayment **P**rotocol]). Ce Serveur_Card utilise sa propre base de données BD_CARD, qui contient essentiellement les comptes et les numéros de cartes (un compte peut avoir plusieurs cartes de crédit associées).

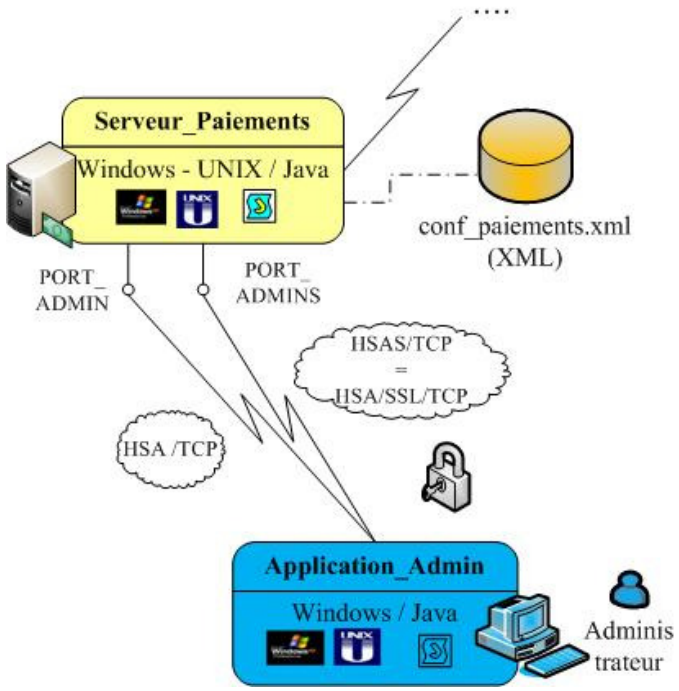
Dans la vraie vie, le compte est en réalité hébergé sur serveur propre à chaque banque : c'est en fait ce serveur qui règle les débits/crédits. Dans ce cas, Serveur_Card ne dispose plus que des numéros de cartes et contacte le serveur de la banque concernée pour finaliser (ou refuser) le paiement des réservations. Cette communication, réglée par un protocole **PARP** [**P**ayment **A**greement or **R**efusal **P**rotocol) qu'il vous appartient de définir et d'implémenter, **est intégralement chiffrée et authentifiée en utilisant le sous-protocole SSL**.

Pour faciliter les choses en fonction du temps de développement disponible, nous considérerons qu'il n'existe qu'un seul **Serveur_Banque** qui utilise sa propre base de données BD_COMPTES.



4.2 Une administration sécurisée

On se souviendra que le serveur **Serveur_Paiements** (notamment) est administrable par une application **Application_Admin** permettant les tâches d'administration à distance avec un protocole HSA (commandes LOGINA, LCLIENTS, PAUSE, STOP).
Ces commandes passent en clair sur le réseau, ce qui peut être ressenti comme dangereux dans certains cas. C'est pourquoi il est demandé de doter l'Application_Admin de la possibilité d'accéder au serveur par une voie de communication sécurisée par SSL. Elle utilise en fait dans ce cas un protocole HSAS (l'analogue de HTTPS) en contactant le serveur sur un port PORT_ADMINS - ces ports seront définis dans le fichier de configuration du serveur (voir ci-dessous).



5. XML: un fichier de configuration

Le fichier de configuration de **Serveur_Paiements** n'est plus un fichier properties mais un fichier XML avec une DTD associée. En s'inspirant du server.xml de Tomcat, on trouve dans ce fichier comme balises principales

- <noms serveurs> : nom officiel, non DNS, etc
- <infos root> : informations sur l'administration du serveur
- <connector> : les ports utilisés par Application_Admin selon qu'il communique en clair ou avec SSL (dans ce cas, il faut aussi, notamment, le chemin des keystores);
- <architecture> : le fait d'être en pool de threads ou pas ainsi que le nombre maximum de threads soit démarrés dans le pools de threads associé ou soit acceptés à la demande;
- <databases> : une série de <database> avec le nom de la base, celui de son SGBD et sa version, etc
- et toute autre balise permettant de paramétrer au mieux le serveur.

Le fichier XML est parsé au démarrage du serveur. En cas d'erreur de conformité avec la DTD ou d'erreur de parsing, le serveur ne démarre pas.

Bon travail !

s: CV & CC