Cahier des charges

Site: Université de Reims Champagne-Ardenne
Cours: Modélisation client-serveur et programmation Web avancée
Livre: Cahier des charges

Imprimé par: SAMI DRIOUCHE
Date: dimanche 11 décembre 2022, 23:26

Description

Description du projet et des attendus associés.

Table des matières

1. Ce qui sera à rendre

Comme cela a été évoqué dans le sujet de TP 1, votre projet concerne la mise en oeuvre d'un système d'achat d'énergie (avec quelques simplifications) qui va être l'occasion de faire dialoguer différentes entités selon un principe client/serveur. Le contexte est donc celui du TP 1.

Ce projet demandera deux rendus rassemblés au sein d'une même archive .zip (de moins de 50Mo) :

1. un rapport décrivant le projet et incluant tout ce qui a été demandé (voir page "contenu du rapport"). Une version PDF ainsi qu'une **version source** doivent être présentes (docx, odt, tex, ...).
2. le code source avec les moyens pour facilement reproduire un scénario de fonctionnement type (ces scénarios sont évoqués dans le chapitre "contenu du rapport").

Vous obtiendrez donc 2 notes qui respecteront le barème (voir page "barème").

2. Contenu du rapport

Vous allez produire un rapport qui devra déjà avoir la forme d'un "vrai" rapport :

- page de garde
- sommaire (automatique)
- en-tête/pied-de-page
- ...

Dans ce rapport vous devrez inclure :

1. un schéma global détaillant toutes les entités en présence et les technologies de communication associées entre elles (on ne demande pas un diagramme de classe). Vous devrez en outre préciser dans quel ordre doit être démarré toutes ces entités pour que le système soit fonctionnel (même si dans votre projet, tout sera peut-être automatisé).
2. Pour chaque couple d'entités pouvant communiquer entre elles, il faudra indiquer :
    - le format des messages échangés et la justification de ce choix de format,
    - la liste des requêtes associées,
    - le détail d'un de ces échanges pour illustrer son fonctionnement (c'est ici qu'il faudrait mettre un chronogramme si vous les faites en TD mais pour maintenant des schémas informels suffiront).
3. Pour les scénarios suivants, établir les échanges complets qui ont lieu entre toutes les entités concernées :
    - Scénario A : dans un système ayant un seul PONE et un seul TARE (en plus du revendeur, du marché de gros et de l'AMI), le client demande une quantité d'énergie sans aucune contrainte particulière et sa demande est toute de suite satisfaite car le PONE produit exactement le type d'énergie demandé. Il y a donc achat (avec enregistrement de l'achat côté AMI) puis distribution au revendeur (en passant par le TARE).
    - Scénario B : dans un système ayant un seul PONE et un seul TARE (en plus du revendeur, du marché de gros et de l'AMI), le client demande une quantité d'énergie qui n'est pas disponible pour le moment ... Par contre, au bout de quelques instants de fonctionnement, la demande du client peut-être satisfaite (car le PONE a suffisamment fourni d'énergie). A voir si cela est possible par une nouvelle demande du client ou si le

revendeur (et/ou le TARE) est capable de suivre l'évolution du marché de gros pour savoir si la demande peut être satisfaite (la première solution est plus simple que la seconde).

- Scénario C : Dans un système ayant un seul PONE et un seul TARE (en plus du revendeur, du marché de gros et de l'AMI), la demande du client ne peut être satisfaite à cause d'une contrainte que le PONE ne peut satisfaire. Par contre, pendant que le système fonctionne, un PONE est rajouté et il peut fournir l'énergie demandée (en une seule production ou au bout d'un certain nombre de production). L'achat d'énergie devient possible et le reste du déroulement respecte le modèle classique.

- Scénario D : Dans un système ayant deux PONEs et deux TAREs (en plus du revendeur, du marché de gros et de l'AMI), pour être satisfaite rapidement, la demande du client doit se baser sur la sommes des énergies fournies par les 2 PONEs. Par contre, les TAREs ne peuvent acheter qu'à un seul lot d'énergie d'un PONE à la fois (via le marché de gros). C'est donc le revendeur qui doit gérer le rassemblement des 2 retours de TARE pour construire la réponse au client.

- Scénario A2 : il s'agit du scénario A dans lequel, à la fin de l'achat de l'énergie, le client demande à faire vérifier le numéro de suivi de l'énergie achetée. Cela signifie qu'il faut valider le CRACHA et le CRADO qu'il contient auprès de l'AMI qui est l'autorité de certification.

En plus de ces éléments, vous pourrez expliquer les choix particuliers d'implémentation que vous avez pu prendre (ce qui sort de ce que vous avez vu en TP).

3. Architecture SALE

Dans ce chapitre, nous allons détailler les communications que vous allez devoir utiliser entre les différents entités et aussi intégrer où le chiffrement intervient (TP 7 et 8).

Serveurs du système SALE

Ainsi, vous aurez de base :

- un unique serveur PHP qui sera utilisé pour le site du revendeur. C'est sur ce site que l'utilisateur pourra consulter les énergies disponibles et préparer sa commande (et voir l'état d'avancement de cette dernière si cette fonctionnalité est proposée).
- un serveur HTTP JAVA pour chaque TARE présent dans le système (un pour commencer, cela irait très bien). Ainsi, tout requête du revendeur nécessitant d'interroger un TARE partira du site PHP vers le serveur HTTP JAVA concerné.
- un serveur UDP pour le marché de gros qui peut non seulement répondre aux requêtes des TARE (demande d'information sur les énergies ou achat d'un lot d'énergie) mais aussi recevoir les envois de lots d'énergie en provenance des PONEs.
- un serveur TCP AMI qui permet de vérifier que tout se passe bien sur le marché de gros. Le marche de gros est le client du serveur AMI et communique avec celui-ci pour faire valider les prix de vente des lots d'énergie proposés par les PONEs et valider les ventes faites avec des TAREs.

Cette architecture vous permettra d'initier tous les traitements demandés.

A cette architecture va se rajouter une partie sécurité en 2 points :

- tout interaction entre serveur/client JAVA devra être chiffrée symétriquement AES ou asymétrique RSA;
- tout lot d'énergie se verra accompagné d'un CRADO (voir sujet de TP 8) lors de sa production puis d'un CRACHA (voir sujet de TP 8) lors sa vente.

4. Mise en oeuvre SALE

Comme vous pouvez vous en douter, le premier écueil pour mettre en oeuvre votre programme sera de gérer tous les serveurs qui doivent tourner ensemble. Pour faciliter (en centraliser) la gestion de l'ensemble (sauf partir PHP), vous allez faire en sorte que tout fonctionne (sauf PHP) au sein d'un même terminal. Ainsi, au lieu de lancer plein de "main", vous allez plutôt lancer des Threads qui s'occuperont de lancer vos serveurs. Il faudra donc garder la main dessus car il faudra être capable d'apporter des modifications au système pendant son fonctionnement.

Le problème d'un unique terminal est la gestion de l'affichage. Pour que tout ne soit pas trop bouillon, chaque entité qui veut faire un affichage dans votre terminal devra le faire en préfixant son message de son identité. Par exemple, si un PONE envoie un lot d'énergie au marché de gros, deux messages vont apparaître dans le terminal :

[PONE]: J'ai fourni le lot d'énergie suivant *(ici détail du lot)*.
[Marche de Gros] : J'ai reçu le lot d'énergie *(ici même détail du lot)*.

Pour vous aider à mettre cela en place, je vous propose un petit exemple que j'ai construit autour de certains fiches de TPs. En gros, le principe est assez simple : quand vous avez terminé de tester des classes avec des mains classiques (et plusieurs terminaux), vous transformez votre classe avec un main en classe classique qui implémente l'interface **java.lang.Runnable** et dont la méthode *main* devient tout simplement la méthode *run*. De plus, vous pouvez y ajouter une instance d'un petit objet *Messenger* que j'ai fait pour gérer l'affichage des noms des objets (mais vous pouvez gérer cela vous même au besoin.

Enfin, il y a beaucoup de paramètres à gérer donc il faut pouvoir aisément s'y retrouver sans créer de conflits. Le plus simple est d'établir un (ou plusieurs) fichier(s) de configuration. A une certaine époque, il y avait un TP en JAVA avec la bibliothèque json.org à ce propos en INFO0503 mais le temps manque maintenant. Par conséquent, je vous donne accès directement à une classe qui est capable de deux choses :

- générer un fichier de configuration à partir d'une 1ère configuration réalisée en dur dans le code. On ne fait généralement cette action qu'une seule fois pour générer le fichier de configuration initial.
- lire un fichier de configuration et donner accès à tous les champs qu'il contient.

Elle vous sera pour tout établir correctement.

Au final, vous avez donc accès à 3 archives (incluant la documentation associée) :

1. Messenger.zip : explique comment fonctionne l'objet Messenger qui permet de préfixer des messages sur la sortie standard
2. Configuration.zip : démonstration de l'objet Configuration qui permet d'utiliser des fichiers JSON pour gérer la configuration d'un système.

3. Lanceur.zip : mise en oeuvre de la fiche 1 du TP 6 sous forme d'un unique main. On remarquera que cet exemple utilise les 2 objets précédemment cités.

5. Barème

Dans ce projet, il y a beaucoup de chose à mettre en place. Votre objectif principal est de prouver que vous maîtrisez un ensemble de technologies de communication et leur bonne coordination. C'est pour cela que le barème exposé dans ce chapitre valorise l'aspect fonctionnel de votre système d'une manière simple à une manière plus évoluée.

Barème du rapport

Voici le détail du barème actuel (qui pourra être amené à évoluer). Pour le moment, c'est un barème sur 72 points répartis ainsi :

- Mise en forme global (4 points) : nombre de page minimum de 10, page de garde, table des matières automatiques, en-tête/pied-de-page, ... Le tout peut être complément annulé par un malus de 4 points si le rapport est rempli de fautes (orthographe/grammaire) et/ou si la gestion de la composition des éléments est trop souvent en dur et à la main (retour chariot excédentaires, ...). Au passage, le malus peut aussi être acté sans que de la mise en forme soit faite (ce qui fait perdre au final 8 points).
- Présentation globale (8 points) : cela évalue le schéma global du système SALE et le détail du démarrage d'un tel système.
- Pour chaque type de communication, il faut détailler le format des échanges, la liste des requêtes associées et au moins un détail (ou chronogramme si les TDs amènent cette compétence) d'exemple (voir chapitre "contenu du rapport"). Cela rapporte 12 points par partie et il y en a 4 : revendeur/TARE, TARE/marché de gros, PONE/marché de gros, marché de gros/AMI. Cela fait donc un total de 48 points.
- Pour les scénarios d'exécution (voir chapitre "contenu du rapport"), chaque détail du déroulement des échanges (idéalement chronogramme) rapporte 4 points (et il y en a 5 donc 20 points).

Barème du projet (code)

Voici le détail du barème actuel (il peut être amené à évoluer). Pour le moment, c'est un barème sur 130 points (+10 bonus) répartis ainsi :

- Gestion du système (6 points) : il s'agit de la présence d'une aide concrète au démarrage du système SALE. Normalement, ces points sont acquis si vous partez du système proposé de tout démarrer dans une seule classe. La gestion par fichier de configuration est aussi une aide non négligeable. Une aide à la mise en oeuvre des scénarios serait un plus.
- Les différentes liaisons entre les entités :

- Serveur PHP concessionnaire (6 points) : est évalué ici le bon fonctionnement du serveur aux demandes du client ainsi que la bonne connexion avec le(s) TARE(s);
- Serveur HTTP TARE (6 points) : est évalué ici le bon fonctionnement du serveur aux requêtes du revendeur ainsi que la bonne connexion avec le marché de gros;
- Liaison UDP TARE/marché de gros (6 points) : est évalué ici le bon fonctionnement du serveur et du(des) clients associés ;
- Liaison UDP PONE/marché de gros (6 points) : est évalué ici le bon fonctionnement du serveur et du(des) clients associés ;
- Liaison TCP marché de gros / AMI (6 points) : est évalué ici le bon fonctionnement du serveur et du(des) clients associés.

- Le bon fonctionnement des scénarios : le A rapporte 5 points, le B 10 points, le C 15 points, le D 20 points et le A2 10 points bonus.
- La sécurité peut rapporter un maximum de 20 points :
  - toutes les communications (JAVA) sont chiffrées :
    - en AES : 5 points
    - ou en RSA : 10 points (si vous faites les CRADO/CRACHA cela vaut le coup de faire du RSA aussi pour les communications).
  - les CRADO sont présents (création et vérification) : 5 points
  - les CRACHA sont présents (création et vérification) : 5 points
- Le rendu des différents TP (2 points extensible à 4 points donc) : si les rendus sont réguliers et correspondent globalement au demandé (format + contenu), les 2 points sont obtenables facilement. Les 2 points supplémentaires (en bonus sur la note sur 20) sont récupérables si chaque rendu a répondu complètement à chaque demande associée.

Remarque

Notez que pour tous ces éléments, il y a toujours 5 états possibles : non fait, à peine fait/fonctionnel, partiellement fait/fonctionnel, presque fait/fonctionnelle et fait/fonctionnel qui rapportent respectivement 0% / 25% / 50% / 75% / 100% des points associés.

Bon courage.

Enjoy ;-)

Jean-Charles BOISSON