# PROGRAMMATION SYSTEME ET RESEAUX

## MINI-PROJET

## VENET AUX ENCHERES


## Résumé

Ce projet de développement a pour objectif la mise en place d'un système informatique. Ce système doit fournir un ensemble de fonctions (Partie 1) et doit être architecturé selon le modèle client-serveur (Partie 2). Il doit gérer des informations multiples de manière persistante (Partie 3) et doit utiliser un protocole de communication bien défini entre les clients et le serveur (Partie 4).



## Table des matières

# 1. Fonctionnement : Vente aux enchères

L'objectif de ce projet est de mettre au point un système centralisé de vente aux enchères d'objets. En effet, la vente aux enchères est l'une des plus anciennes techniques de vente. Actuellement, de nombreux particuliers et professionnels font appel à ce système de vente pour se débarrasser de certains objets afin de pouvoir obtenir de la liquidité pour financer un nouveau projet ou tout simplement de rembourser quelques dettes. Les ventes aux enchères sont ouvertes à toute personne majeure.

Préalablement, le commissaire-priseur se charge d'estimer le prix de départ de chaque bien. Par la suite, la vente aux enchères sera annoncée afin d'attirer le plus grand nombre d'acheteurs potentiels. Le jour J, le commissaire-priseur présentera tous les objets disponibles à la vente et annoncera la mise à prix de départ. Par la suite, les personnes désireuses d'acquérir un objet devront enchérir le bien et proposer chacun son prix. La vente se clôturera dès lors que le commissaire-priseur a annoncé le dernier prix et qu'aucun autre acheteur n'a proposé un nouveau prix au bout de 30 secondes. Le commissaire-priseur abat alors son marteau et le dernier acheteur ayant énoncé le prix le plus élevé se verra attribué l'objet.

Par la suite, il devra aller récupérer son bien et le payer immédiatement, car il n'existe pas de délai de réflexion ou de rétractation dans les ventes aux enchères. De plus, le prix de vente sera augmenté d'environ 20 % selon le montant des frais à payer en supplément.

## 1.1. Données manipulées

### a) Données des objets mis aux enchères

Le système informatique dispose d'un ensemble de données où sont décrits tous les biens mis aux enchères par le commissaire-priseur. Pour chaque bien, on dispose des informations suivantes :

- la référence précise de l'objet;
- le prix de départ ;
- le dernier prix qui reflète à chaque instant la dernière valeur proposée par un acheteur ;
- l'état qui indique si le bien a été vendu ou non
- l'acheteur qui a emporté l'objet si la vente est conclue

Tout cet ensemble de données est enregistrée sous la forme d'un fichier texte **"bien.txt"** et ce selon le format suivant :

| Référence Objet | Prix Départ | Dernier Prix | Etat | Acheteur |
|---|---|---|---|---|

Un exemple de contenu du fichier **"bien.txt"** est :

| Référence Objet | Prix Départ | Dernier Prix | Etat | Acheteur |
|---|---|---|---|---|
| 1 | 150 | 1200 | Vendu | 1 |
| 2 | 800 | 800 | Disponible | 0 |
| 3 | 250 | 500 | Vendu | 2 |
| 4 | 500 | 500 | Vendu | 1 |

Ce fichier doit être mis à jour au niveau du dernier prix à chaque fois où un acheteur annonce un prix supérieur au précédent et si la vente est conclue alors l'état va changer de **Disponible** à **Vendu**. En plus, initialement le champ acheteur est mis à zéro pour tous les objets.

**N.B.** Le Dernier prix d'un bien est initialisé au prix de départ et ensuite il sera augmenté à travers les propositions des acheteurs. Une fois le bien est vendu, ce prix ne sera pas modifié.

### b) Données des séances de vente

Les données relatives aux propositions sont sauvegardées dans un fichier "**histo.txt**". Ce fichier permet d'assurer la traçabilité des différentes propositions reçues sur un bien donné. Il est structuré selon le format suivant :

| Identificateur Acheteur | Proposition Valeur | Résultat |
|---|---|---|

Un exemple du fichier "**histo.txt**" relatif au **bien 1** est le suivant :

| Identificateur Acheteur | Proposition Valeur | Résultat |
|---|---|---|
| 1 | 200 | echec |
| 2 | 300 | echec |
| 1 | 400 | echec |
| 2 | 450 | echec |
| 3 | 1000 | echec |
| 2 | 1200 | succes |

Le bien 1 son prix de départ est 150 (d'après le fichier **"bien.txt"**). L'acheteur 1 a commencé par proposer le prix 200. Ensuite, l'acheteur 2 a proposé la valeur 300. L'acheteur 1 repropose une deuxième valeur qui est 400. Le deuxième acheteur repropose aussi une autre valeur de 450. Un nouvel acheteur 3 vient ensuite proposer la valeur 1000. Finalement, l'acheteur 1 propose une valeur de 1200.

Ce fichier "**histo.txt**" est remplie par les propositions selon un ordre chronologique et doit être mis à jour à chaque fois où une proposition d'un acheteur est reçue.

### c) Données des factures

Le système informatique dispose aussi d'un fichier qui décrit la somme que les acheteurs devront payer. Ce fichier contient pour chaque acheteur :

- L'identificateur de l'acheteur ;
- La somme totale à payer par cet acheteur.

Ces factures sont sauvegardées dans un fichier texte **"facture.txt"** et ce selon le format suivant :

| Identificateur Client | Somme à payer |
|---|---|

Un exemple de contenu du fichier **"facture.txt"** est :

| Identificateur Acheteur | Somme à payer |
|---|---|
| 1 | 2040 |
| 2 | 600 |
| 3 | 0 |

L'acheteur 1 a acheté le bien 1 et le bien 4 (d'après le fichier **"bien.txt"**). La somme à payer est donc 1200 + 500 à laquelle il faut ajouter 20% de frais. Dans ce sens aussi, l'acheteur 2 va payer 500*(1+20%) = 600. Alors que l'acheteur 3 ne va rien payer puisqu'il n'a rien acheté.

Dans le cadre de ce projet, on ne demande pas le détail des différentes factures. On ne s'occupe que par la somme totale que l'acheteur doit payer au commissaire-priseur.

### 1.2. Opérations possibles sur les données

Le système informatique doit permettre de réaliser les opérations suivantes :

Pour le commissaire-priseur :

- **Consulter la liste des biens :** en donnant la référence d'un bien, on doit pouvoir récupérer les informations le concernant (prix de départ, dernier prix, état et l'identificateur de son acheteur s'il est vendu).
- **Consulter la facture d'un acheteur :** il doit être possible de voir la facture correspondant à un acheteur en précisant son identificateur.
- **Consulter l'historique des proposions** : il doit être possible de voir le contenu de l'historique des propositions sur un bien donné.

Pour l'acheteur :

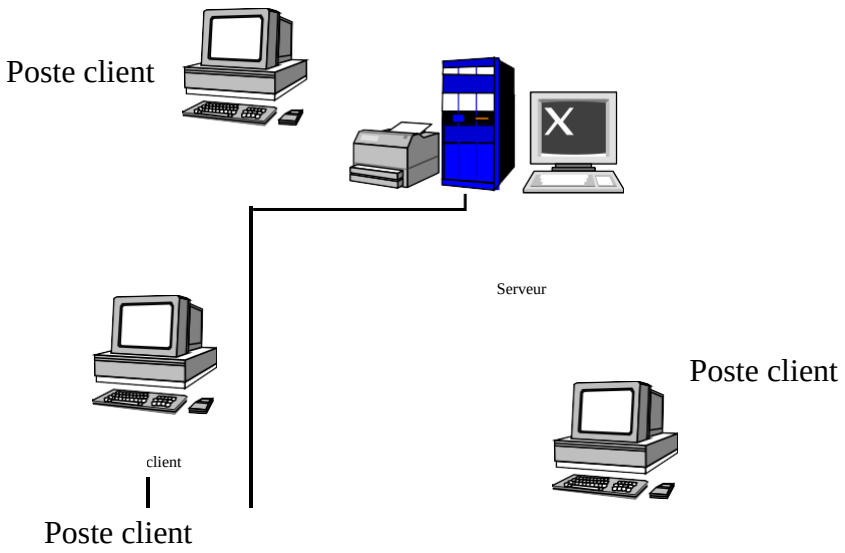
- **Acheter un bien :** un acheteur doit pouvoir acheter un bien disponible. Il doit être capable de proposer une valeur pour un bien mis aux enchères ;
- **Recevoir une facture :** un acheteur doit recevoir une facture à chaque fois qu'il a achevé sa participation aux enchères.

## 2. Architecture du système informatisé

Le commissaire-priseur souhaite mettre en place un système informatique de vente des enchères en ligne composé des éléments suivants :

- un serveur central;
- des postes clients répartis chez les acheteurs.

Les postes clients doivent permettre, grâce à une Interface Homme-Machine appropriée (qui sera simulée ici sous forme d'une zone de saisie en mode caractères), de réaliser les différentes opérations prévues.



Le fonctionnement sera le suivant :

1. Le serveur annonce le lancement d'une vente aux enchères d'un bien donné et donc il doit spécifier sa référence et son prix de départ
2. Selon la proposition de l'acheteur, le poste client préparera une requête à envoyer au serveur.
3. La requête sera envoyée au serveur et le client se mettra en attente de la réponse.
4. Le serveur réceptionnera la requête et la traitera pour comprendre la demande du client.
5. Il effectuera ensuite le traitement associé,
6. Le serveur enverra le résultat de ce traitement aux différents postes clients si le prix du bien a été mis à jour.
7. Le client réceptionnera le résultat et pourra enchaîner sur une nouvelle requête.

## 3. Gestion des données

Le serveur central aura à sa charge la gestion des ventes et donc de l'ensemble des données (biens, propositions et facturation). Les données manipulées par le serveur sont sauvegardées dans des fichiers. La concurrence d'accès au fichier exige l'emploi d'un mécanisme de synchronisation.

## 4. Gestion des échanges client/serveur

Les échanges entre les clients et le serveur doivent suivre un protocole bien défini pour que le serveur comprenne les requêtes des clients et pour que les clients comprennent les résultats renvoyés par le serveur. Le protocole à adopter dans ce cas de projet est TCP et ce pour assurer une communication en mode connecté et mettre en *temps réel* le contenu des différents fichiers.

## 5. Consignes

Le projet est travaillé et étudié en **quadrinôme** mais la notation est **individuelle**.

### 5.1. Travail de base demandé

Vous disposez d'un squelette de programme fournie au niveau du cours. Après lecture des différents documents, il faudra le compléter pour fournir :

1. **Aspect réseau :** Protocole TCP ; Connexion entre au moins entre **trois** machines physiques ;
2. **Aspect système :** un serveur parallèle. Il s'agit de modifier le serveur TCP pour qu'il puisse gérer des requêtes en parallèle. Le serveur doit donc créer un nouveau thread chaque fois qu'il reçoit une requête ;
3. **Aspect programmation :** une sauvegarde/rechargement des données dans des fichiers. Cela permet que les informations soient persistantes et non pas limitées à la session en cours.

## 5.2. Travaux complémentaires

Après la réalisation du travail demandé dans la section précédente, vous pourrez choisir un ou plusieurs points présentés ici pour améliorer votre programme.

1. **Aspect réseau :** Faites en sorte que votre serveur et votre client puissent fonctionner indifféremment en TCP ou en UDP. Le protocole TCP ou UDP sera choisi selon la valeur d'un argument de la ligne de commande ;
2. **Evolution fonctionnelle :** Introduisez la notion de profil en distinguant un profil administrateur d'un profil utilisateur. Désormais, l'administrateur sera le seul à avoir le droit d'effectuer une requête listant les utilisateurs et récupérant les données.

Dans tous les cas, avant de commencer à coder ces questions, faites une analyse des problèmes que vous voulez résoudre et des solutions que vous allez proposer. Si cette analyse est bien faite, le codage sera facile. Une bonne analyse (même sans implémentation) dans le rapport et la soutenance sera fortement valorisée.

## 5.3. Organisation du travail

Les groupes peuvent travailler au **maximum** en **quadrinôme**. La répartition des tâches entre les quatre est libre. Nous vous conseillons de :

- bien respecter le cahier de charges ;
- bien gérer le temps qui vous est imparti ;
- bien discuter dans le groupe ;
- réfléchir avant de programmer.

Pour la réalisation logicielle, nous conseillons vivement de suivre les étapes suivantes :

1. définir les fichiers *.txt;
2. réalisation d'une communication en TCP de chaque côté (client et serveur) ;
3. réalisation des opérations possibles de chaque côté (client et serveur) ;
4. ajout du mode parallèle (thread) ;
5. ajout de la synchronisation ;
6. ajout des fonctionnalités supplémentaires.

**Note** : certaines étapes peuvent être faites en parallèle. Pensez à vous répartir le travail.

## 5.4. Évaluation du travail

### a) Soutenance

Chaque groupe aura **20 minutes** pour présenter son travail. Le groupe expliquera brièvement la répartition du travail entre ses membres, la réalisation du projet et les problèmes rencontrés. Des questions seront posées et la démonstration est évidemment demandée.

### b) Rapport

Le groupe doit rendre juste avant la soutenance un rapport de 10 pages au maximum donnant:

- l'organisation du travail dans le groupe ;
- la méthodologie utilisée dans le développement ;
- la pertinence de certains choix ;
- l'état courant du projet ;

- les difficultés rencontrées ;
- un rapide bilan de ce que vous a apporté ce projet.

Il s'agit de mettre en valeur la qualité de votre travail à l'aide d'un rapport. Pour cela le rapport doit explicitement faire le point sur les fonctionnalités du logiciel (lister les objectifs atteints, lister ce qui ne fonctionne pas et expliquer - autant que possible - pourquoi).

Ensuite le rapport doit mettre en valeur le travail réalisé sans paraphraser le code, bien au contraire : il s'agit de rendre explicite ce que ne montre pas le code, de démontrer que le code produit a fait l'objet d'un travail réfléchi et même parfois minutieux. Par exemple, on pourra évoquer comment vous avez su résoudre un bug, comment vous avez su éviter/éliminer des redondances dans votre code, comment vous avez su contourner une difficulté technique ou encore expliquer vos choix, pourquoi certaines pistes examinées voire réalisées ont été abandonnées.

Il s'agira aussi de bien préciser l'origine de tout texte ou toute portion de code empruntée (sur internet, par exemple) ou réalisée en collaboration avec tout autre groupe. Il est évident que tout manque de sincérité sera lourdement sanctionné.

Le rapport ne doit pas redonner des informations présentes dans ce document.

### c) Code source

Vous devrez également fournir l'intégralité de votre code source en **Python** sous le format d'un dossier compressé qui doit avoir un nom selon le modèle suivant :

**Groupe**_Nom1_ Nom2_ Nom3_Nom4

Le groupe indique votre section : A, B, C ou D.

N'oubliez pas de commenter vos programmes. L'archive doit être nettoyée i.e. ne doit pas contenir des fichiers objets ou des exécutables.

Le dossier compressé est à charger sur un lien Filepiper qui vous sera communiqué. Le mot de passe est **enicarthage**.