RFC – Projet Cbay

Lorrain BALBIANI Manavai TEIKITUHAAHAA

Table des matières

Introduction

Dans le cadre d'un projet de programmation en C dans notre cursus d'ingénieur T.R.I, on est amenés à programmer une architecture Serveur/Client d'une vente aux enchères.

Cette RFC (Request For Comments) décrit les échanges entre le serveur et le client et permet ainsi une programmation indépendante des deux entités.

Formalisme

Requête du client : ——————>
Réponse du serveur :
Normal : ----------->
Erreur : ----------->

Les mots dans une requête ou une réponse présentant un « $ » en début de mot est une variable.

Exemple :

$USER signifie que le nom d'utilisateur devra être remplacé dans la requête à la position de cette variable.

$MODE signifie le mode dans lequel un utilisateur est connecté.

En Annexe, vous trouverez les structures définissant les paramètres des clients et du serveur, ainsi qu'une liste répertoriée de toutes les erreurs qui pourront être envoyées par le serveur en fonction des requêtes.

Les codes d'erreur qui seront en fin de RFC, sont au format hexadécimal suivant :

- 0x000
- 0x001

Un utilisateur standard désigne un utilisateur qui n'est pas Administrateur.

Le terme «doit » désigne l'obligation d'une action qui devra être effectuée par les développeurs du client ou du serveur.

Connexion au serveur

La connexion au serveur se fait sur un socket TCP/IP sur un numéro de port décidé par les développeurs de l'application.

Le serveur ne gère qu'une seule connexion à la fois pour faciliter la programmation et pour éviter toute concurrence sur un objet qui causera des erreurs non-prises en compte dans cette RFC.

Structures de donnée

Les structures de donnée citée ci-dessous peuvent être modifiées en conséquence selon les besoins, les fonctionnalités ou les optimisations recherchées par les développeurs.
En bref ces structures ne sont là que pour inspirer les programmeurs.

Voici la structure de donnée minimum définissant un client (utilisateur de l'application cliente) :

```
     struct user_t
{
     unique_id_t uid;           // IDENTIFIANT UNIQUE
     char login[21];            // NOM D'UTILISATEUR
     char password[21];         // MOT DE PASSE
     bool admin;                // MODE ADMIN(TRUE or FALSE)
}
```

Voici la structure de donnée minimum définissant un objet :

```
  struct object_t
  {
     unique_id_t uid;           // IDENTIFIANT UNIQUE DE L'ITEM
     char name[51];             //  NOM  DE  L'OBJET  (à  l'unité)
exemple: pour un lot de XXX: "XXX"

     char category[51];         //  CATEGORIE  (mobilier,  électro,
auto, vêtement, livre, )

     char description[1024];    // DESCRIPTION DE L'OBJET OU DU LOT
MIS AUX ENCHERES

     char url_image[100];       // URL IMAGE
     float start_price;         // PRIX DE DEPART DE L'ENCHERE
     float temp_price;          // PRIX INTERMEDIAIRES (prix proposé
durant l'enchère par les utilisateurs, le prix affiché est celui le
plus élevé de la mise)

     float final_price;         // PRIX FINAL DE L'ENCHERE
     int quantity;              //  QUANTITE  (le  prix  de  l'enchère
est pour tout le lot)

     char place[50];            // ADRESSE DE L'OBJETS
     unique_id_t vendeur;       // UID DU VENDEUR
     unique_id_t acheteur;      // UID DE L'ACHETEUR (initialisé à 0
/ mis à 0 lors de la suppresion d'un utilisateur lors d'un enchère)

  };
```

Attention, la taille des variables dans les requêtes ou réponse doit être au maximum égale à la taille prévu dans les structures de données de cette RFC. Par exemple, un nom d'utilisateur ne devra pas comporter plus de 20 caractères.

Structure des requêtes

Les requêtes seront tous de la forme suivante :

```
REQ_MOBILE_DE_LA_REQUÊTE = $ARGUMENT1 FOR ARGUMENT2 BY ARGUMENT3 \n
```

Les requêtes sont tous délimitées par un mot clé au début et terminées par un « \n ». Le mot clé ne doit pas comporter d'espace entre les mots, mais des tirets (Under-score « _ »). Le séparateur suivant le mot clé de la requête est le signe égal « = » avant d'être suivi par une suite d'argument et de séparateur qui peuvent être :

- Le mot « FOR » ;
- Le mot « BY » ;
- Le signe « + » ;
- Un espace.

Attention, chaque mot de la requête doit être obligatoirement séparé par un espace (sauf le mot clé).

Une requête doit être suivie par une réponse du serveur. La réponse est du même format, c'est-à-dire qu'il doit comporter au minimum un mot clé et doit être délimitée par un « \n » en fin de phrase.

Exemple :

`REQ_CONNECTION = $LOGIN + $PASSWORD \n` est suivi de `USER_CONNECTED = $UID \n.`

Liste des variables

$LOGIN	nom d'utilisateur envoyé ou reçu.
$PASSWORD	mot de passe utilisateur qui sera envoyé dans la requête.
$UID	Identifiant Unique (UID) de l'utilisateur
$MODE	mode de l'utilisateur (acheteur, vendeur ou administrateur).
$ADMIN.UID	UID de l'administrateur
$ADMIN.LOGIN	Nom d'utilisateur de l'administrateur.
$NEW_PW	Nouveau mot de passe envoyé au serveur pour un utilisateur
$OPERATION	Opération de va subir un objet ou utulisateur
$ITEM	Désigne l'objet des enchères. $ITEM.XXX signifie l'accès aux champs XXX dans la requêtes
$MSG_ERROR	Code d'erreur renvoyé par le serveur en cas de problème

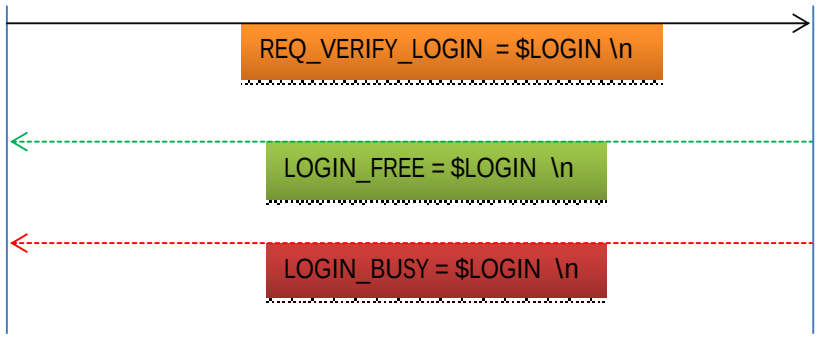
Plusieurs variables ne sont pas lister dans ce tableau. Elles sont cependant évidentes à comprendre.

I. Inscription d'un nouvel utilisateur

Une fois que le programme client est connecté au serveur, celui-ci aura le choix de pouvoir inscrire un nouvel utilisateur avant l'authentification d'un utilisateur. Cette inscription se fait en deux étapes, la vérification de la disponibilité d'un nom d'utilisateur et son inscription.

1. Vérification de la disponibilité d'un nom d'utilisateur.

CLIENT SERVEUR

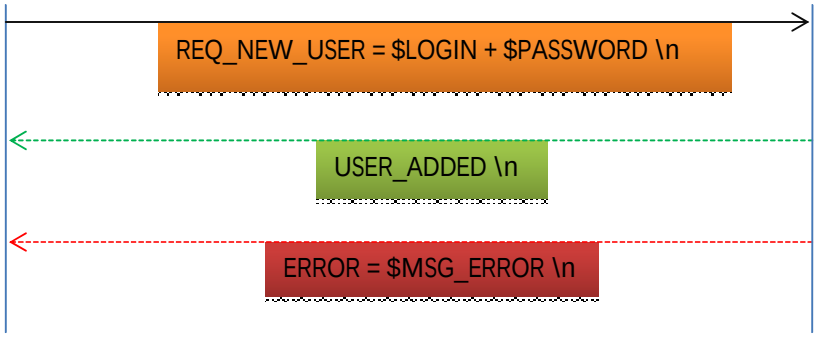


Cette requête permet de vérifier la disponibilité d'un nom d'utilisateur et de prévenir les erreurs lors de l'inscription d'un utilisateur sur le serveur d'enchère.

Si le nom d'utilisateur est disponible, alors le serveur renverra une réponse positive « LOGIN_FREE » ; sinon le serveur renverra une réponse négative « LOGIN_BUSY ».

2. Inscription d'un nouvel utilisateur

CLIENT SERVEUR



Cette requête permet d'inscrire un nouvel utilisateur sur le serveur d'enchère. Il doit comporter un nom d'utilisateur disponible, vérifié au préalable, et un mot de passe qui sont envoyés au serveur.

Une identité unique (UID) est alors créée et correspond à la date d'inscription au format d'heure POSIX (soit le nombre de seconde écoulé depuis le 1er janvier 1970 00:00:00).

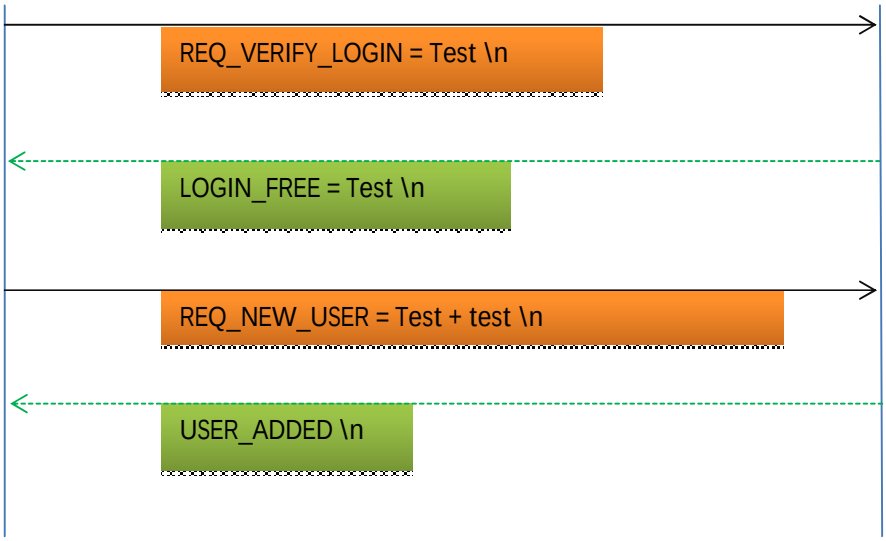
Après traitement par le serveur, celui-ci envoi une réponse positive, « USER_ADDED » ; sinon, il envoie un code d'erreur au format hexadécimal.

3. Exemple

Dans cet exemple, un utilisateur veut vérifier la disponibilité de son nom d'utilisateur avant de tenter une inscription.

Voici les paramètres de l'exemple :

- Nom d'utilisateur : Test
- Mot de passe : Test



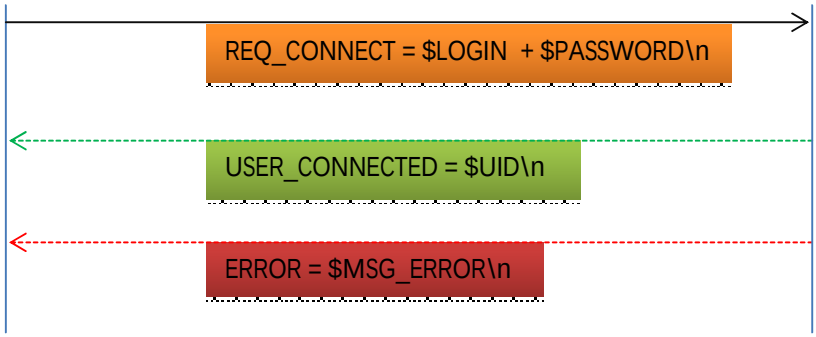
On suppose que dans cet exemple que le nom d'utilisateur était libre. Cependant, s'il ne l'est pas, le client doit revérifier la disponibilité d'un nom d'utilisateur alternatif et ce jusqu'à ce qu'il soit validé par le serveur, avant d'envoyer une requête d'inscription. Si le nom d'utilisateur n'est pas libre lors de l'inscription, le serveur renverra une erreur.

II. Connexion

1. Connexion d'un utilisateur

$LOGIN : nom d'utilisateur qui sera envoyé dans la requête.
$PASSWORD : mot de passe utilisateur qui sera envoyé dans la requête.
$UID : Identifiant unique de l'utilisateur.
$MSG_ERROR : Code d'erreur renvoyé par le serveur en cas de problème.

CLIENT SERVEUR



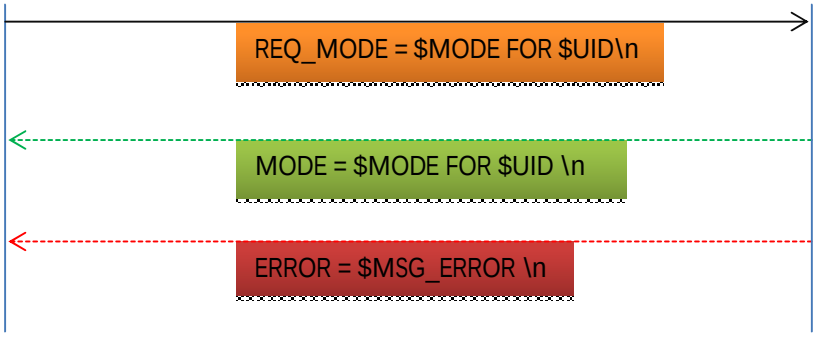
Cette requête permet la connexion d'un utilisateur au serveur d'enchère.

Une requête comportant un nom d'utilisateur disponible et un mot de passe est envoyée au serveur. Si le serveur accepte la connexion après la vérification des identifiants transmis dans la requête, le serveur renvoie l'identifiant unique (UID) de l'utilisateur qui vient de se connecter.

Cette UID permettra l'échange ultérieur entre le client et le serveur des requêtes qui comportera obligatoirement l'UID.

2. Choix du mode

CLIENT SERVEUR



Cette requête permet de choisir un mode utilisateur. Il existe trois modes différents :

- Administrateur : permet la gestion des utilisateurs et des objets aux enchères.
- Acheteur : permet à un utilisateur de mettre de nouveaux objets aux enchères, de visualiser ceux actuellement aux enchères ou un historique de ses ventes.
- Vendeur : ce mode permet à un utilisateur d'enchérir sur des objets. Il pourra aussi avoir une visualisation de ses achats.

Attention : Un utilisateur qui vient de s'inscrire sera le seul à pouvoir choisir le mode administrateur et pourra par la suite revenir sur ce mode s'il le souhaite tandis qu'un utilisateur standard n'aura pas le droit de choisir le mode Administrateur même s'il est proposé au niveau du programme client.

Format de $MODE :

- ADMIN
- BUY
- SELL

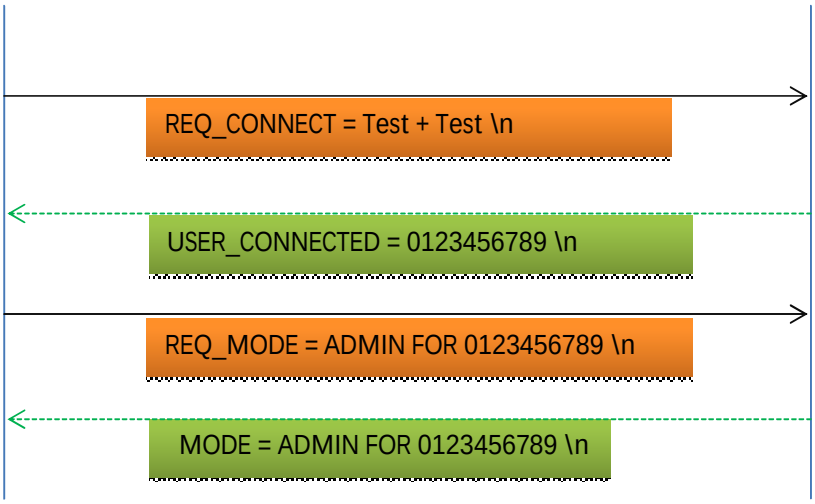
3. Exemples

Dans cet exemple, un utilisateur tente de se connecter au serveur et choisit son mode. On supposera que tout se passe bien.

Paramètres de l'exemple :

- Nom d'utilisateur : Test
- Mot de passe : Test
- Mode : ADMIN

Client Server



III. Mode administrateur

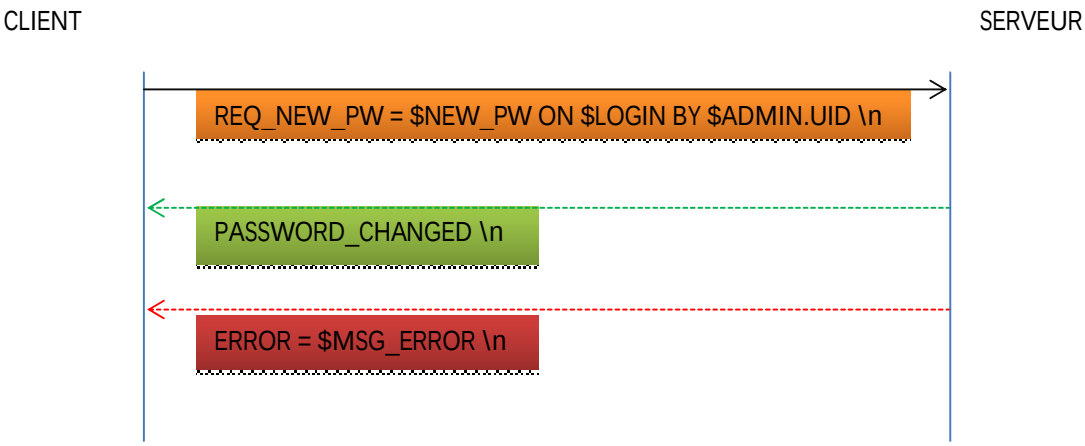
L'Administrateur a les droits suivant :

- Gérer les utilisateurs
- Gérer les ventes (supprimer/annuler une vente)

1. Gestion des utilisateurs

Un administrateur a le droit de soit modifier le mot de passe d'un utilisateur, soit en supprimer un. Lorsqu'un utilisateur est supprimé, tous les objets mis en enchères seront supprimés. Si l'utilisateur a placé des enchères sur un objet alors les enchères seront remises automatiquement à zéro.

a) Changement de mot de passe d'un utilisateur

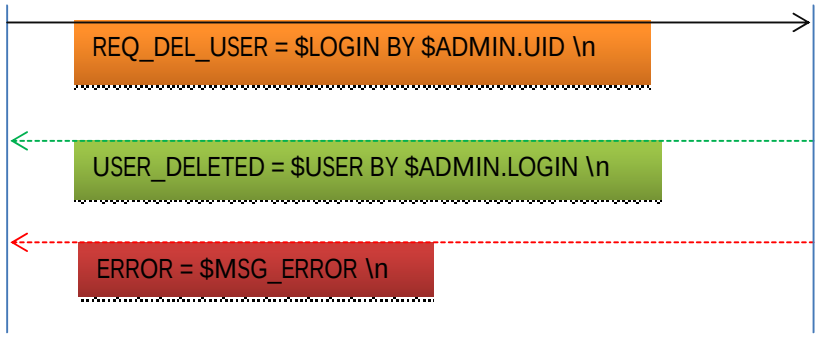


Un administrateur qui désire changer le mot de passe d'un utilisateur doit connaitre le nom de ce dernier. Ainsi dans la requête de modification du mot de passe, le nouveau mot de passe ainsi que le nom d'utilisateur à modifier seront envoyés. L'UID de l'administrateur ne servant qu'à authentifier la requêter.

Si la suppression n'a pas abouti, un message d'erreur au format hexadécimal sera envoyé au client.

b) Suppression d'un utilisateur

CLIENT SERVEUR



Une requête de suppression d'utilisateur ne peut être établie que par un administrateur.

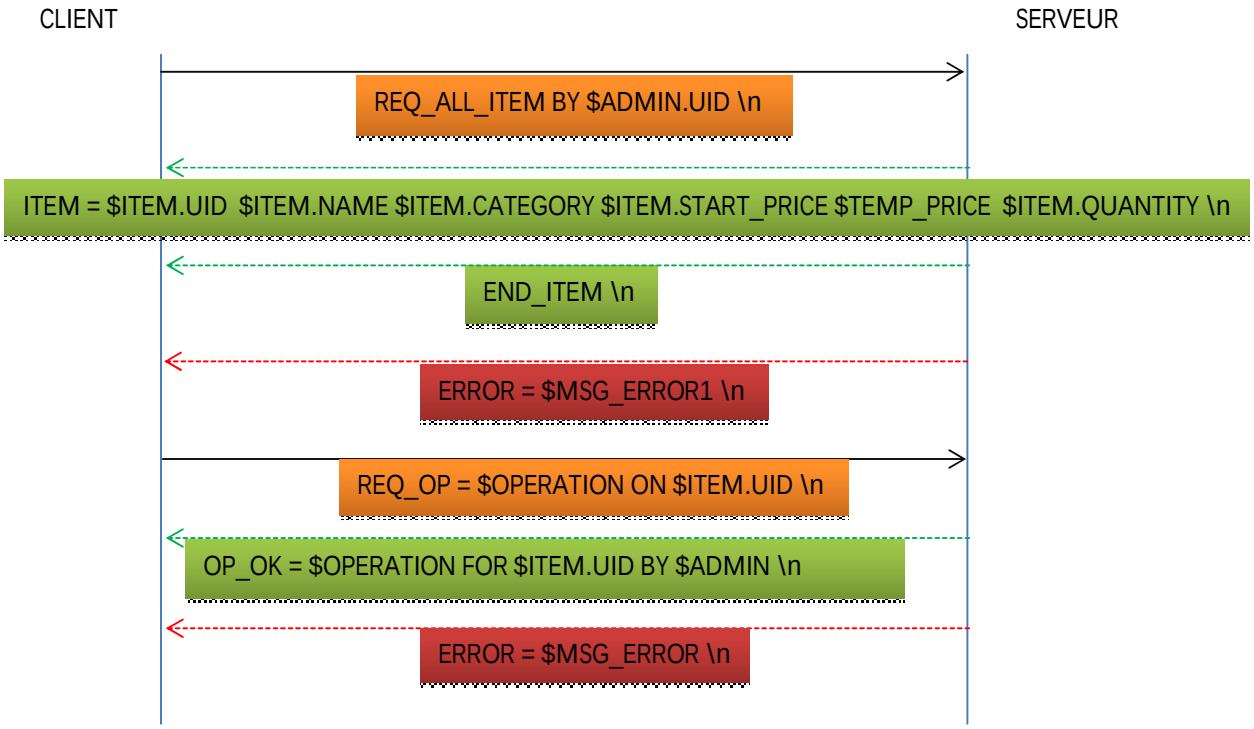
Ainsi seul le nom d'utilisateur qui sera supprimé sera envoyé dans la requête, suivi de l'UID de l'administrateur pour authentifier la requête.

Si la suppression n'a pas abouti, un message d'erreur au format hexadécimal sera envoyé au client.

2. Gestion des objets aux enchères

Dans la gestion des objets, un administrateur recevra l'affichage des tous les objets qui sont aux enchères. L'affichage des objets est à la charge des développeurs de l'application cliente.

Le serveur enverra les objets en fonction de leur date de mise aux enchères.



Une fois l'affichage terminé le client administrateur pourra effectuer ce type d'opération sur un objet :

Format $OPERATION

CANCEL : Supprimer une enchère en cours.

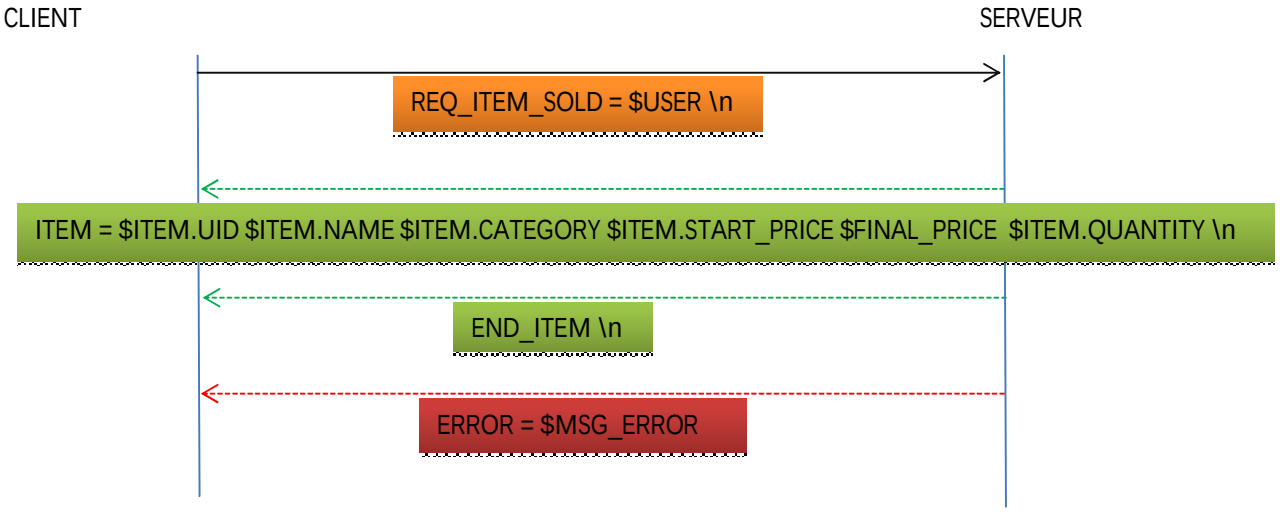
DELETE : Annuler une enchère, c'est-à-dire qu'elle sera remise à zéro et que toutes les offres ayant été effectuées depuis le début seront supprimées.

Si l'une des opérations n'a pas pu aboutir, un message d'erreur au format hexadécimal sera envoyé au client.

IV. Mode vendeur

Ce mode permet aux utilisateurs de pouvoir mettre des objets aux enchères. Il permet aussi d'afficher les objets que l'utilisateur a mis aux enchères et de pouvoir les gérer. Ce mode permettra à l'utilisateur de consulter son historique de vente.

1. Affichage des ventes effectuées



Lorsqu'un utilisateur désire consulter son historique de vente, le client envoie une requête à l'application serveur avec son identifiant unique. Ensuite un algorithme de recherche effectue une recherche sur les objets en faisant la correspondance entre l'UID du vendeur d'un objet et le prix final (il faut que le prix final soit différent de 0 pour que l'objet soit considéré comme vendu).

Les caractéristiques d'un objet sont envoyées au fur et à mesure de la recherche, et lorsqu'il n'y pas plus de correspondance, le serveur envoi au client l'information « END_ITEM ».

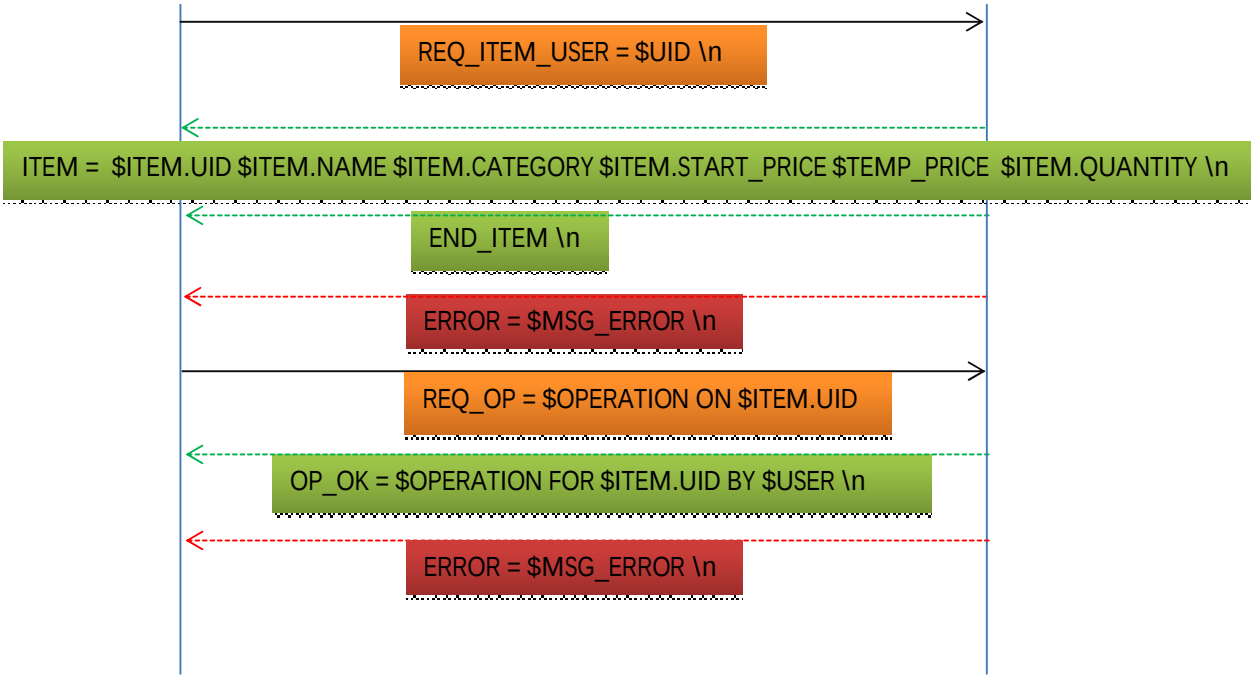
L'affichage des objets est à la charge du développeur de l'application cliente.

2. Gestion des ventes en cours

CLIENT SERVEUR

Le gestionnaire des ventes en cours permet de :

- Afficher les objets en cours d'enchère où vous avez effectué une mise.
- Supprimer une enchère / un objet
- Annuler une enchère.



Lorsque l'utilisateur envoie une requête avec son UID d'utilisateur, le serveur effectue un algorithme de recherche sur l'objet en se basant sur l'UID de l'utilisateur, et le champ FINAL_PRICE d'un objet non-vendu qui doit être nul.

A chaque correspondance, le serveur envoie les caractéristiques de l'objet et quand il n'y a plus de correspondance, le serveur envoie l'information « END_ITEM ».

La gestion de l'affichage est à la charge du développeur de l'application cliente.

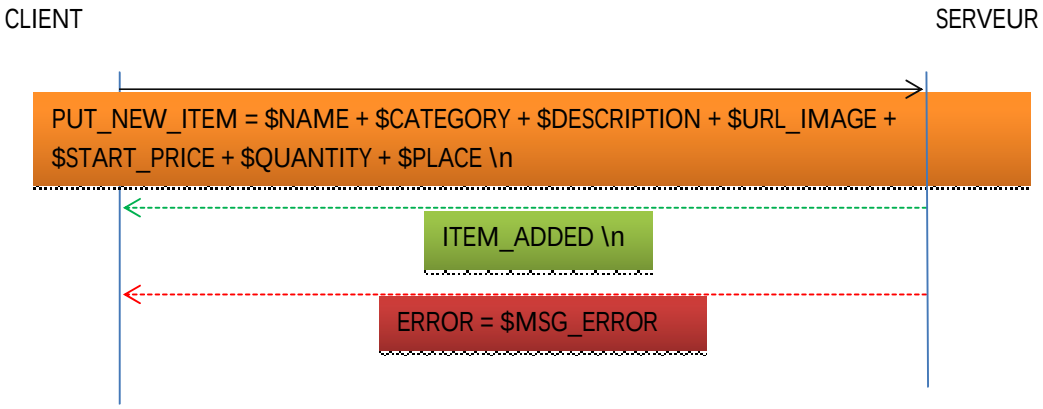
Ensuite l'utilisateur peut sélectionner un objet et y effectuer les opérations citées ci-dessous :

$OPERATION :

DELETE_BID : Permet de supprimer une enchère en cours. L'objet sera supprimé du serveur.

CANCEL_BID : Permet d'annuler une enchère. Tous les compteurs de l'objet seront remis à zéro.

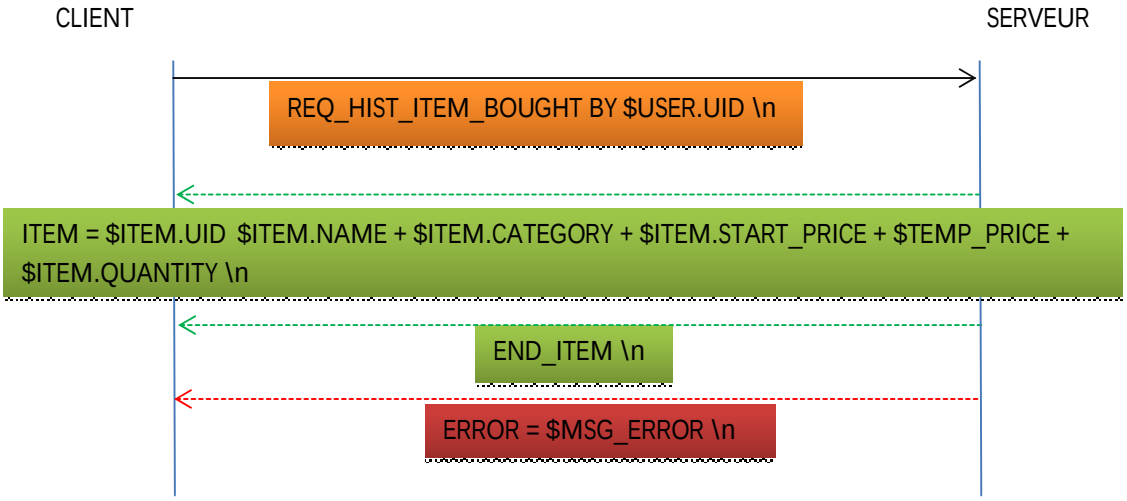
3. Ajout d'un nouvel item



V. Mode Acheteur

Le mode acheteur permet de mettre des enchères sur des objets après les avoir consultés dans une liste de catégorie. Par ailleurs l'Acheteur a la fonctionnalité de pouvoir regarder ses historiques d'achats, de gérer ses enchères, et d'y effectuer des opérations (augmenter une mise, annuler une mise, etc.). Enfin il pourra regarder son historique d'achat.

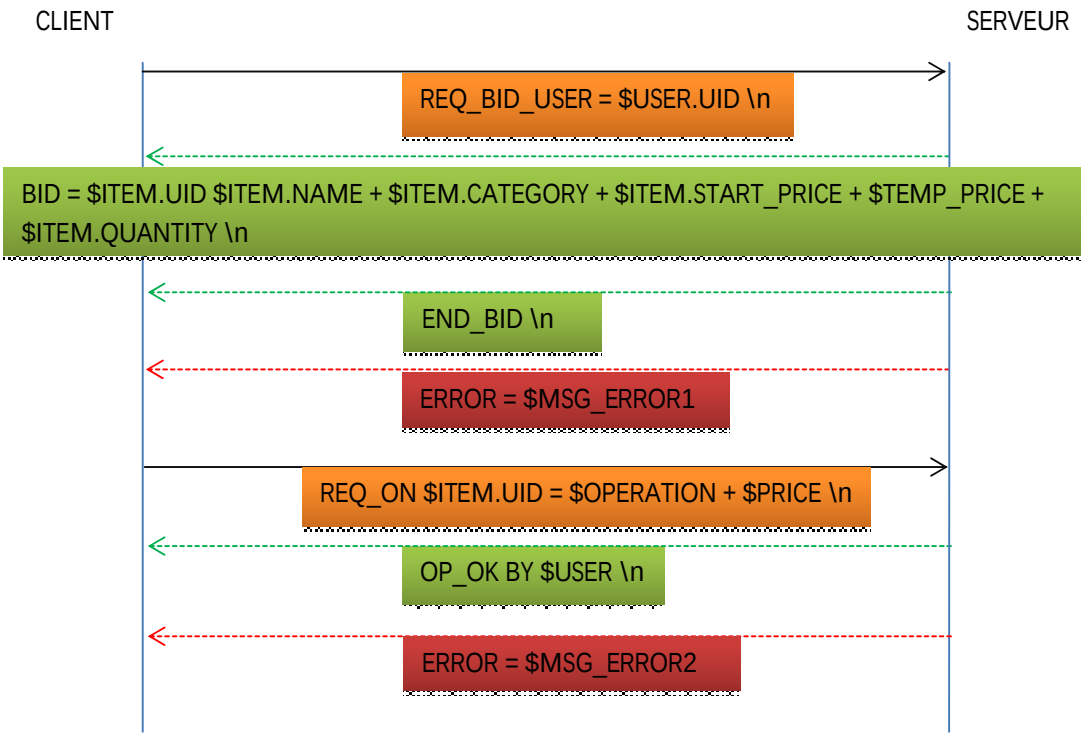
1. Affichage historique des achats



Lorsqu'un utilisateur désire consulter son historique d'achat il envoie une requête à l'application serveur avec son identifiant unique. Ensuite un algorithme de recherche effectue une recherche sur les objets en faisant la correspondance entre l'UID de l'acheteur final d'un objet et l'UID de l'utilisateur. Au fur et à mesure de la correspondance, l'application serveur envoie au client les caractéristiques ci-dessus d'un objet. Lorsqu'il n'y pas plus de correspondance, le serveur envoie au client l'information « END_ITEM » pour signifier qu'il n'y a plus d'objet à afficher.

L'affichage des objets est à la charge du développeur de l'application cliente.

2. Gestion enchères en cours



Le gestionnaire des enchères en cours permet d' :

- Afficher les objets en cours d'enchère où vous avez effectué une mise.
- Augmenter une mise sur un objet ;
- Annuler une enchère.

Lorsque l'utilisateur envoie une requête avec son UID d'utilisateur, le serveur effectue un algorithme de recherche sur les objets en se basant sur un fichier de log (ou de journalisation) qui répertorie toutes les opérations qui se sont passées sur le serveur. Ainsi, l'algorithme fait une correspondance entre l'utilisateur et l'objet. A chaque correspondance, le serveur envoie les caractéristiques de l'objet et quand il n'y a plus de correspondance, le serveur envoie l'information « END_BID ». La gestion de l'affichage est à la charge du développeur de l'application cliente. De plus seules les enchères encore en cours seront affichées. Les enchères terminées ne seront pas consultables par l'utilisateur s'il n'est pas l'acheteur final.

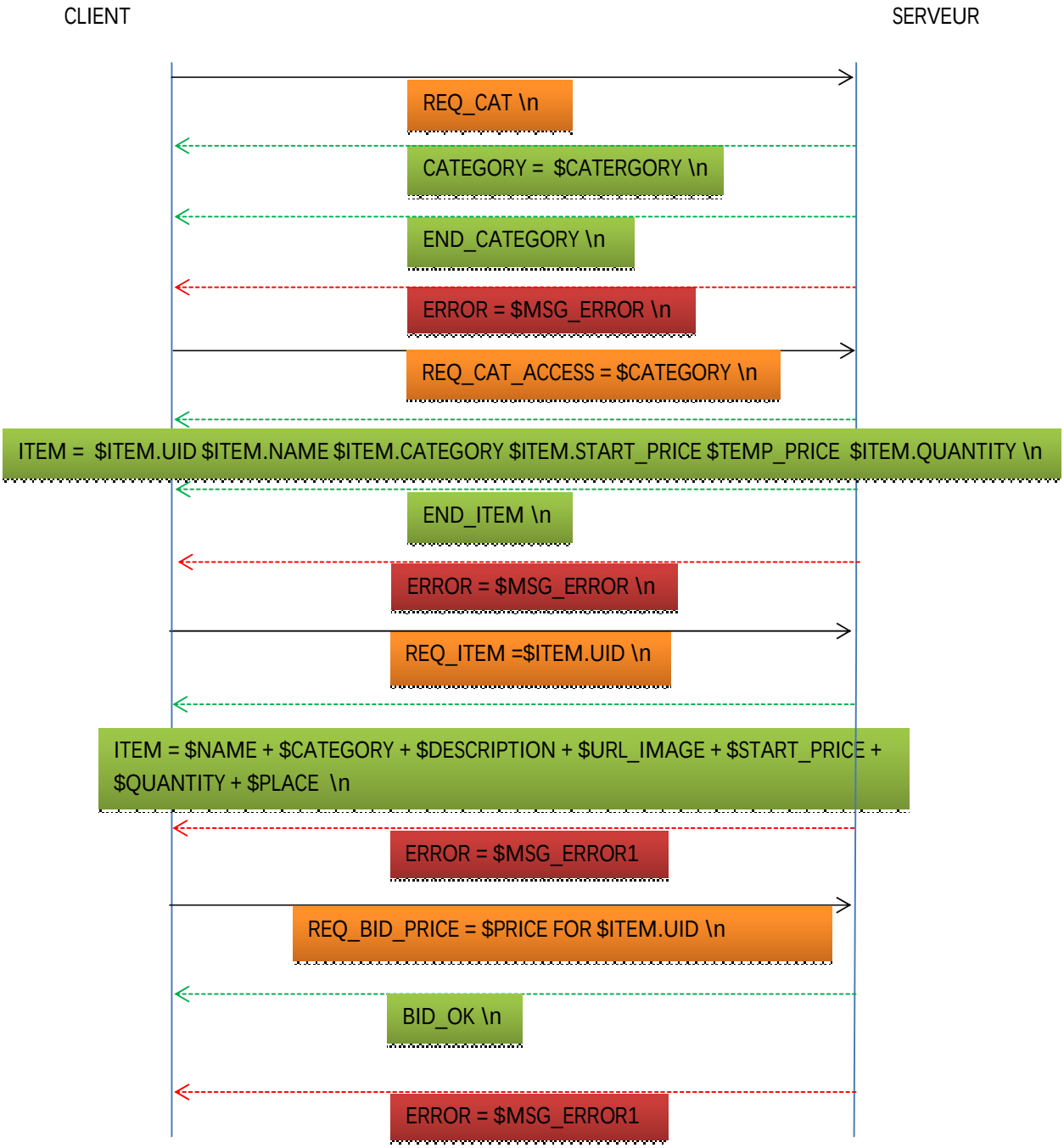
Ensuite l'utilisateur peut sélectionner un objet et y effectuer les opérations citées ci-dessous :

$OPERATION :

UP_BID : Permet d'augmenter une enchère. Cette opération est doit être suivi d'un prix supérieur à la mise en cours.

CANCEL_BID : Permet d'annuler une enchère. Le prix inclus dans la requête doit être nul ou négatif.

3. Enchère sur un nouvel item



Dans ce processus d'enchère sur un nouvel objet, on peut distinguer plusieurs étapes. Une première étape consistant à afficher les catégories disponibles sur l'application serveur. Cette première étape n'est pas obligatoire lorsqu'on a défini des catégories sur les deux applications serveur et cliente. Il s'agit donc ici d'une option offerte au cas où les développeurs voudraient offrir plus de choix dans la sélection des objets.

Les catégories seront envoyées une par une jusqu'à la réception de l'information END_CATEGORY ».

Ensuite, une fois la catégorie choisie par la requête « REQ_CAT_ACCESS », la liste des objets situés dans cette catégories seront envoyé une par une jusqu'à la réception de l'information « END_ITEM ».

Une fois l'objet sélectionné par la requête « REQ_ITEM » en fournissant l'UID de celui-ci, Toutes les informations concernant l'objet seront envoyés. L'affichage est à la charge du développeur de l'application cliente.

Enfin, l'utilisateur pourra s'il le désire effectuer une mise sur l'objet. Cette mise doit être supérieure ou égale au prix de départ s'il s'agit de la première mise. Sinon, la mise doit être supérieure à la mise en cours de l'objet. Il est à la charge de l'application d'effectuer des vérifications préalables lors d'une mise, sinon le serveur enverra un message d'erreur.

VI. Codes d'erreur

Des erreurs pourront être rajoutées, modifiées ou supprimées lors de la programmation des applications par les développeurs. Les erreurs citées ci-dessous ne sont là que pour inspirer les développeurs.

```
//  Code d'erreur lors de l'inscription
```

0x000 "Unknown Error"
0x001 "Login is not free"
0x002 "Login too long"
0x003 "Password too long"

```
//Code D'erreur lors de la connexion d'un utilisateur
```

0x010 "Unknown Error"
0x011 "Login not found"
0x012 "Bad Password"
0x015 "You are actually Banned !"

```
// Codes d'erreur lors du choix du mode
```

0x020 "Unknown Error"
0x021 "You can not choose admin mode"
0x020 "Bad mode: not existing (admin, sell, buy)

```
// Mode Administrateur
```

```
//Erreurs liées au changement de mot de passe d'un utilisateur.
```

0x040 "You are not and Admin"
0x041 "Can not modify password: User not found"
0x042 "Can not modify password: User is an admin"
0x043 "Can not modify password: New password is the same"

```
//Erreurs liées à la suppression d'un utilisateur.
```

0x051 "Can not delete user: User not found"
0x052 "Can not delete user: User is an Admin"
0x054 "Can not delete user: You are not an admin"
0x055 "Can not delete user: Unknown Error"

```
// Mode Vendeur
```

0x060 "Unknown Error"
0x061 "Can not load your selled Items"
0x062 "Can not load your current bid"

0x064 "Can not add new item"

```
// Mode Acheteur
```

0x070 "Unknown Error"
0x071 "Can not load category..."
0x072 "Can not make a bid..."
0x074 "Cant not load hisroty"
0x075 "Bad price: Lower that current bid"

```
// Erreur général
```

0x999 "Internal Server Error"

```
ALGORITHME mainServer
DECLARATION
      Client : Structure de type user_t
      /* Structure défini dans la RFC */
      Buffer : de type chaine de caractère.
      /* données envoyé par le client */
DEBUT
      Chargement des paramètres du server
      Chargement des connexions
      TANT QUE (1)  //BOUCLE INFINIE
      DEBUT TQ
      Attente de connexion d'un client
            TAN QUE (client est connecté)
            DEBUT TQ

            RECEPTIONsurSOCKET(buffer)
/* Pour chaque condition, on appellera une fonction qui va extraire la
requête du buffer et la traiter */
            SI buffer CONTIENT « REQ_VERIFY_LOGIN »           ALORS
                  req_verify_login(client, buffer)
            SINON SI buffer CONTIENT «REQ_NEW_USER »          ALORS
                  Req_new_user(&client, buffer)
            SINON SI buffer CONTIENT «REQ_CONNECT »           ALORS
                  req_connect( &client, buffer)
            SINON SI buffer CONTIENT « REQ_MODE »             ALORS
                  req_mode(&client, buffer)
            SINON SI buffer CONTIENT « REQ_NEW_PW »           ALORS
                  req_new_pw(&client, buffer)
            SINON SI buffer CONTIENT « REQ_DEL_USER »         ALORS
                  req_del_user(&client, buffer)
            SINON SI buffer CONTIENT « REQ_ALL_ITEM »         ALORS
                  req_all_item(&client, buffer)
            SINON SI buffer CONTIENT « REQ_OP »               ALORS
                  req_op(&client, buffer)
            SINON SI buffer CONTIENT « REQ_ITEM_SOLD »        ALORS
                  req_item_sold(&client, buffer)
            SINON SI buffer CONTIENT « REQ_ITEM_USER »        ALORS
                  req_item_user(&client, buffer)
            SINON SI buffer CONTIENT « REQ_HIST_ITEM_BOUGHT » ALORS
                  req_hist_item_bought(&client, buffer)
            SINON SI buffer CONTIENT « REQ_BID_USER »         ALORS
                  req_bid_user(&client, buffer)
            SINON SI buffer CONTIENT  « REQ_ON »              ALORS
                  req_on(&client, buffer)
            SINON SI buffer CONTIENT « REQ_CAT »              ALORS
                  req_cat(&client)
            SINON SI buffer CONTIENT « REQ_CAT_ACCESS »       ALORS
                  req_cat_access(&client, buffer)
            SINON SI buffer CONTIENT « REQ_ITEM »             ALORS
                  req_item(&client, buffer)
            SINON SI buffer CONTIENT « REQ_BID_PRICE »        ALORS
                  req_bid_price(&client, buffer)

            FIN TQ
      FIN TQ
FIN
```

```
ALGORITHME mainClient
DECLARATION
      mode : mode de l'utilisateur
      choix : choix en fonction des modes
DEBUT

      Connexion au serveur
      SI utilisateur N'A PAS DE compte ALORS
            Inscription d'un nouvel utilisateur
      Connexion de l'utilisateur
      Choix du mode
/*************************  MODE ADMIN  ****************************/
      SI mode EST « ADMIN » ALORS
      DEBUTSI
            FAIRE TQ (choix != Quitter le Programme)
            DEBUT FTQ
                  choix :
                        Gestion des utilisateurs
                        Gestion des objets
                        Quitter le Programme
            FIN FTQ
      FINSI
/*************************  MODE VENDEUR ***************************/
      SI mode EST « VENDEUR » ALORS
      DEBUTSI
            FAIRE TQ (choix != Quitter le Programme)
            DEBUT FTQ
                  choix :
                        Historique des ventes
                        Gestion des ventes en cours
                        Ajout d'un nouvel item aux enchères
                        Quitter le Programme
            FIN FTQ
      FINSI
/*************************  MODE ACHETEUR ***************************/
      SI mode EST « ACHETEUR » ALORS
      DEBUTSI
            FAIRE TQ (choix != Quitter le Programme)
            DEBUT FTQ
                  choix :
                        Historique des achats
                        Gestion des enchères en cours
                        Enchère sur un nouvel item
                        Quitter le Programme
            FIN FTQ
      FINSI
FIN

/* Quand l'utilisateur voudra quitter le programme, on fera la fonction
exit() en langage c */
```