PROTOCOLE

```
HELP***                                                    (Client -> Serveur)
WELCOME messageBienvenue***                                (Serveur -> Client)

//messageBienvenue EST UNE CHAÎNE DE CARACTÈRE PRÉDÉFINIE QUE LE SERVEUR
//RENVOIE AU CLIENT QUAND IL INITIALISE LA COMMUNICATION AVEC LE SERVEUR OU
//QUAND L'UTILISATEUR ENVOIE LA REQUÊTE «HELP»
//«WELCOME» EST SUIVI D'UN ESPACE PUIS D'UN SAUT A LA LIGNE
//ENSUITE, CHAQUE EXPLICATION DE COMMANDE DOIT ÊTRE ACHEVÉ PAR UN SAUT DE LIGNE

NEW pseudo motDePasse***                          (Client -> Serveur)
NEW_SUCCESS***                                             (Serveur -> Client)
NEW_ERROR***                                               (Serveur -> Client)

//INTERDICTION D'UTILISER DES ESPACES DANS LE MOT DE PASSE
//UN SEUL PRÉNOM EST AUTORISÉ
//NEW INSCRIT LE CLIENT DANS LA BASE DE DONNÉE MAIS NE LE CONNECTE PAS.
//POUR SE CONNECTER LE CLIENT DOIT UTILISER LA COMMANDE «CONNECT»

CONNECT pseudo motDePasse port***                          (Client -> Serveur)
CONNECT_SUCCESS pseudo adresse port***                     (Serveur -> Client)
CONNECT_ERROR***                                           (Serveur -> Client)

//Le CONNECT_SUCCESS PERMETTRA DE RÉCUPÉRER LES INFORMATIONS DES CHAMPS
//CHEZ LE CLIENT
//CONNECT_ERROR EST RENVOYÉ AU CLIENT PAR LE SERVEUR SI LE PSEUDO N'EXISTE PAS
//SI LE MOT DE PASSE EST INCORRECT OU SI LE PORT CHOISI EST DÉJÀ OCCUPÉ

INFO_DOMAINES***                                           (Client -> Serveur)
LISTE_DOMAINE messageListeDomaine***                       (Serveur -> Client)

//messageListeDomaine EST UNE CHAÎNE DE CARACTÈRE PRÉDÉFINIE QUE LE SERVEUR
//ENVOIE AU CLIENT QUAND L'UTILISATEUR ENVOIE LA COMMANDE «INFO_DOMAINE»
//«LISTE _DOMAINE» DOIT ÊTRE SUIVI D'UN ESPACE PUIS D'UN SAUT A LA LIGNE
//ENSUITE, CHAQUE DOMAINE EST SUIVI D'UN SAUT DE LIGNE

ADD_ANNONCE titreAnnonce||domaine prix description***      (Client -> Serveur)
ADD_SUCCESS***                                             (Serveur -> Client)
ADD_ERROR***                                               (Serveur -> Client)

//LE MOTIF '||' DOIT APPARAÎTRE UNE SEULE FOIS SUR LA COMMANDE ADD_ANNONCES POUR
//SEPARER LE TITRE DES AUTRES CHAMPS
//LA DESCRIPTION PEUT ÊTRE VIDE

DELETE_ANNONCE idAnnonce***                                (Client -> Serveur)
DELETE_SUCCESS***                                          (Serveur -> Client)
DELETE_ERROR***                                            (Serveur -> Client)
ANNONCE_NO_EXIST***                                        (Serveur -> Client)

//SUPPRIME UNE ANNONCE ET RENVOIE DELETE_SUCCESS AU CLIENT
//SI idAnnonce N'EXISTE PAS ON ENVOIE ANNONCE_NO_EXIST
//SI L'UTILISATEUR NE S'EST PAS CONNECTÉ, RENVOIE UNE DELETE_ERROR

CHECK_ALL_ANNONCES***                                      (Client -> Serveur)
//Liste de toutes les annonces de tous les clients

CHECK_ANNONCES_CLIENT pseudoUser***                        (Client -> Serveur)
CLIENT_NOT_EXIST***                                        (Serveur -> Client)

//Liste des annonces d'un client
//Si le pseudo demandé n'existe pas dans la base de donnée
```

```
CHECK_ANNONCES_DOMAINE Domaine***                          (Client -> Serveur)
DOMAINE_NOT_EXIST***                                       (Serveur -> Client)

//Liste des annonces pour un domaine
//Si le domaine demandé n'existe pas, renvoie l’erreur DOMAINE_NOT_EXIST

CHECK_ANNONCES_PRICE prix***                               (Client -> Serveur)
PRICE_ERROR***                                             (Serveur -> Client)
//Liste des annonces inférieures ou égales au prix donné
//Si le prix est mal inséré , par exemple, des lettres au lieu d'un chiffre

//LES COMMANDES PRÉCÉDENTES COMMENÇANT PAR CHECK_ALL OU CHECK_ANNONCES
//REÇOIVENT LA MÊME RÉPONSE DU SERVEUR EN CAS DE SUCCÈS : ANNONCES ET ANNONCE


ANNONCES_NB nombre_annonce***                              (Serveur -> Client)

//Envoie le nombre d'annonces

ANNONCE id domaine prix titreAnnonce pseudo***             (Serveur -> Client)

//Envoie l'annonce sans description,seulement le titre
//La aille du titre doit être comprise entre 4 et 30

CHECK_DESCRIPTION idAnnonce***                             (Client -> Serveur)
ANNONCE_NOT_EXIST***                                       (Serveur -> Client)
DESCRIPTION_VIDE                                           (Serveur -> Client)
DESCRIPTION description***                                 (Serveur -> Client)

//Permet de voir la description complète d’une annonces
//Renvoie une l’erreur ANNONCE_NOT_EXIST si l’identifiant de l’annonce
//n’existe pas
//DESCRIPTION_VIDE n’est pas une erreur, cela indique que la description
//renseignée par la personne a qui appartient est vide

NOPE***                                                    (Serveur -> Client)

//A chaque fois que le client essaie de faire une commande sans s'être connecté

INVALID***                                                 (Serveur -> Client)

//Si le message envoie par l’utilisateur ne corresponds a aucun message du
//protocole ou si le nombre de champs inséré par l’utilisateur ne correspond pas
//au nombre de champs de la commande souhaité

DISCONNECT***                                              (Client -> Serveur)
DISCONNECT_SUCCESS***                                      (Serveur -> Client)
DISCONNECT_ERROR***                                        (Serveur -> Client)

//COUPE LA CONNEXION

QUIT***
                                                           (Client -> Serveur)
//COUPE LA COMMUNICATION

BYE***                                                     (Serveur -> Client)

//CONFIRME LA FIN DE COMMUNICATION
```

Protocole PeerToPeer

WHOIS pseudo*** (Client -> Serveur)
IT_IS adresse port*** (Serveur -> Client)

//WHOIS donne au client l'adresse et le port du pseudo desiré.
//Le serveur renvoie IT_IS en cas de succès, NOPE si le client n'est pas
//connecté et CLIENT_NO_EXIST si le pseudo n'existe pas dans la base de donnée.

CALL_OPEN adresseClient2 portClient2 **monPseudo monAdresse monPort*****
(Client1 -> ServeurClient2)

CALL_OPEN_SUCCESS pseudoClient2*** (ServeurClient2 → Client1)
CALL_OPEN_ERROR*** (ServeurClient2 → Client1)

//Pour CALL_OPEN, L'UTILISATEUR INSÈRE UNIQUEMENT adresseClient2 et portClient2
//Les champs dans la partie en gras soulignée doit être ajouté dans le code
//Le client l'ajoute dans l'entrée de l'utilisateur1 avant de l'envoyer au7
//client2

(Détail d'implémentation →> L'information de ces champs doit donc être récupéré
au préalable du coté client)

CALL **mypseudo** pseudo message*** (Client1 -> ServeurClient2)
SENT*** (ServeurClient2 →> Client1)

//CALL envoie un message au pseudo désigné
//Les erreurs doivent être vérifie avant d'être envoyé au client2
//Si une erreur est détecte (Si, par exemple, on essaie d'utiliser CALL sans
//avoir utilisé CALL_OPEN avec pseudo ou si pseudo n'existe pas),
//ON N'ENVOIE RIEN AU CLIENT2
//mypseudo EST INSÉRÉ PAR LE CLIENT DANS LE CODE (et non pas par
//l'utilisateur)
//Pas de saut de ligne dans le message, dès le moment ou l'utilisateur
//appuie sur la touche « Entrée » , le message est envoyé

CALL_CLOSE **mypseudo** pseudoClient2*** (Client1 -> ServeurClient2)

[CALL_CLOSE mypseudo pseudoClient1***] (ServeurClient2 →> ServeurClient1)
[CALL_CLOSE_OK mypseudo pseudoClient2***] (ServeurClient1 →> Client2)

CALL_CLOSE_OK pseudoCLient2*** (ServeurClient2 →> Client1)

//Comme pour CALL, les erreurs doivent être vérifié
//avant d'être envoyé au client2
//Si une erreur est détecté (Si, par exemple, on essaie d'utiliser CALL sans
//avoir utilisé CALL_OPEN), ON N'ENVOIE RIEN AU CLIENT2
//mypseudo EST INSÉRÉ PAR LE CLIENT DANS LE CODE (et non pas par
//l'utilisateur)

//Le CALL_CLOSE en rouge envoyé par le serveurCLient2 n'est pas une entrée
//de l'utilisateur2, ce message doit être produit (et envoyé au
//ServeurCLient1) dans le code lorsque le serveurCLient2 reçoit le CALL_CLOSE du
//Client (qui, ici, est bien une entrée du l'utilisateur1)
//L'envoi du CALL_CLOSE_OK doit être géré au même endroit et confirme
//la fermeture de la communication