Documentation du projet My_Teams

Fait par Deffontaine Steven, Valtriani Lucka et Bernard Antoine.

Voici le code pour le server.h :

```c
//Structure des data du client
typedef struct {
    char *username;
    char uuid[37];
    int socket;
    set_t state;
    char *team_uuid;
} client_data_t;

//Structure des messages
typedef struct {
    char *sender_uuid;
    char *receiver_uuid;
    char *message;
    int timestamp;
} message_t;

//Structure du serveur
typedef struct server_t server_t;

typedef void (*command_function_t)(server_t *, char **);

typedef struct {
    char *name;
    command_function_t function;
} command_t;
```

Ou nous stockons toute les informations utile pour les clients (Socket, identification, le nom...), nous aurons de même pour les informations sur les messages (le contenu, l'ID de l'envoyeur et du receveur...) et pour finir une structure pour les différentes commandes que nous pourrions taper.

Pour finir dans ce fichier nous aurons la structure principale du fichier « server_t » ou nous aurons donc toute la gestion du server my_teams avec toutes les fonctions nécéssessaire.

```c
typedef void (*log_function_t)(char **);

typedef struct {
    char *name;
    log_function_t function;
} log_t;

struct client_t {
    int socket;
    struct sockaddr_in addr;
    log_t *logs;
    int nbr_logs;
    char *username;
};

void help(void);
void receive_message_handler(client_t *client, char *message);
void liblog_handler(client_t *client, char *message);
void init_liblog(client_t *client);

// Log functions
void loginlog(char **args);
void logoutlog(char **args);
void unknown_user(char **args);
```

Dans le fichier client.h nous aurons les structures principales pour gérer les clients pour avoir leur noms, sockets, username...

Nous avons aussi une liste de commande que les clients peuvent utiliser :

« /help » : affiche de l'aide et une liste de commande disponible.

« /login « user_name » » : connecte l'utilisateur avec le nom spécifié.

« /logout » : déconnecte le client du serveur.

« /users » : récupère la liste de tous les utilisateurs du serveur.

« /user ["user_uuid"] « : Récupère les détails concernant un utilisateur spécifié par son UUID.

« /send ["user_uuid"] ["message_body"] » : Envoie un message à l'utilisateur spécifié par UUID.

« /messages ["user_uuid"] » : Liste tous les messages échangés avec l'utilisateur spécifié.

« /create ["team_name"] ["team_description"] » : Crée une nouvelle équipe avec le nom et la description donnés.

« /subscribe ["team_uuid"] » : Abonne le client aux événements d'une équipe spécifique et de ses sous-répertoires.

« /subscribed ?["team_uuid"] » : Liste toutes les équipes auxquelles l'utilisateur est abonné, ou tous les utilisateurs abonnés à une équipe spécifique.

« /unsubscribe ["team_uuid"] » : Désabonne le client de l'équipe spécifiée

« /use?["team_uuid"]?["channel_uuid"]?["thread_uuid"] » : Définit le contexte de commande à une équipe, un canal ou un fil de discussion spécifique.

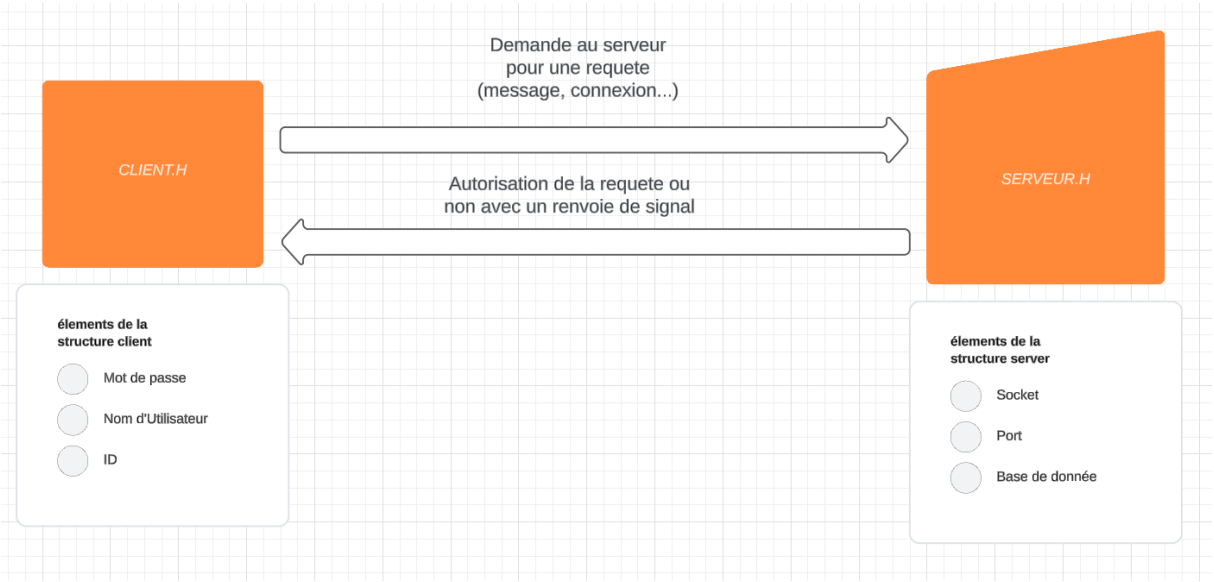
« /create ["channel_name"] ["channel_description"] » : Crée un canal dans l'équipe actuellement sélectionnée.

« /create ["thread title"] ["thread message"] » : Initie un fil de discussion dans le canal actuellement sélectionné.

« /create ["comment_body"] » : Publie un commentaire dans le fil de discussion actuellement sélectionné.

« /list » : Liste les ressources pertinentes au contexte actuel (équipes, canaux, fils de discussion ou réponses).

« /info » : Affiche les détails sur la ressource actuellement en contexte (utilisateur, équipe, canal ou fil de discussion).


Demande au serveur
pour une requete
(message, connexion...)
CLIENT.H
SERVEUR.H
Autorisation de la requete ou
non avec un renvoie de signal
élements de la
structure client
Mot de passe
Nom d'Utilisateur
ID
élements de la
structure server
Socket
Port
Base de donnée