Doc Client UDP:

En rentrant dans la room:
#include “**UDPClient.hpp**”

Appeler le constructeur de `UDPClient(io_context, ip);`
`io_context` et `ip` t’es censé les avoir
Faire **directement** une requête “`INIT_PLAYER name`”
Le `name` est le nom du joueur initialisé au

Comment faire une requête UDP? (this si tu veux update un truc dans ton jeux)

```
client.request(message, [this](std::string response) {
    std::cout << response;
});
```

Il n’y a pas de “`\n`” à la fin des requêtes envoyés/reçues.

Lien vers les routes:
https://docs.google.com/drawings/d/1CUQjSUeGGLv6YmwZHSJHLmAV0R8fU3mAsjxOG0K3COs/edit

Exemple pour récupérer toutes les **positions**.

```
client.request(“GET_POSITIONS”, [this](std::string response) {
    std::cout << response; //
});
```

```
NE PAS OUBLIER DE FAIRE io_context.run();
```

TCP ROUTES

CREATE_ROOM name MaxSlots
JOIN_ROOM RoomName
SET_NAME PlayerName
GET_ROOMS
GET_ROOM_PLAYERS RoomName
LEAVE_ROOM RoomName
MESSAGE msg
GET_MESSAGES

```
tcp.request(command, [res, game]() {
        for (int i = 0; i < res.arguments; i++) {
                game.rooms.emplace_back(res.arg(i));
        }
});
```

UDP ROUTES

Server Route

COLLISION
UDPATE_PLAYER_POS x y
INIT_PLAYER name
MOVE_PLAYER x y
FIRE_BULLET startX startY
DEAD monsterId
UPDATE_SCORE score
READY 0/1
MSG msg

Client Routes (Web hooks)

GAME_START
GET_SCORE
NEW_BULLET x y monster|player
GET_POSITIONS:
- x
- y
- player_id (1 - 4, -1 si non-joueur)
- monster_id (-1 si joueur)
- asset (string of asset)

Routes UDP & TCP