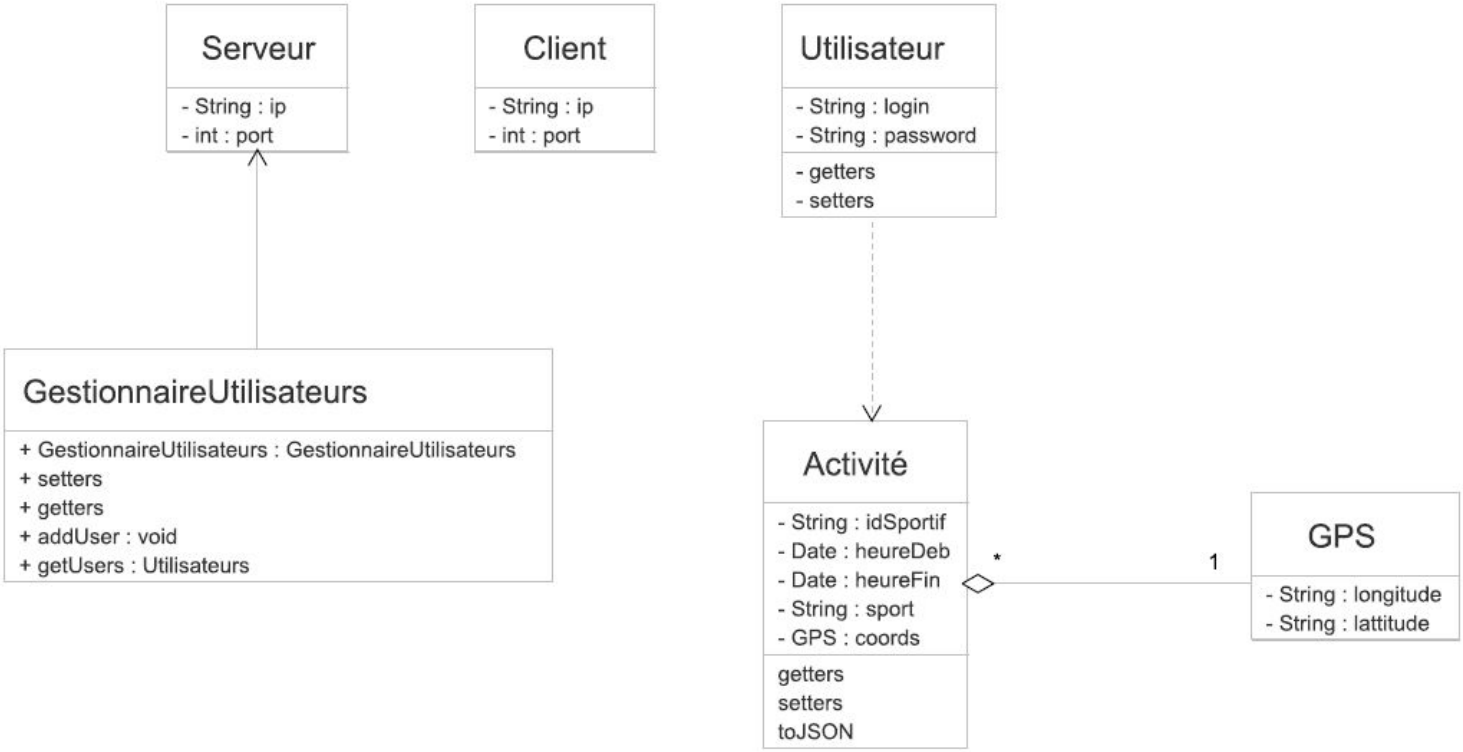
UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE

Rapport de projet INFO0503

Simulation d'une application de tracking sportif

GIGOUT Thomas
DAUNIQUE Wilfried
3ème année de Licence informatique

Diagramme de classe



Le diagramme de classe ci-dessus représente la modélisation pré-programmation.

Nous avons ainsi pu constater nos erreurs, et voir qu'il nous manquait quelques éléments à ce diagramme de classe notamment :

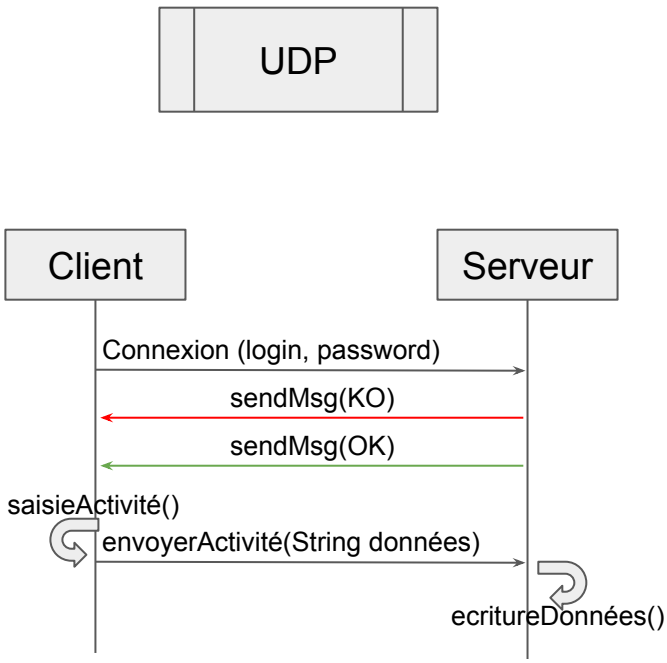
- La classe Config
    - En découle les fonctions creerFichierConfiguration dans Client et Serveur
- Des attributs de type Config dans Client / Serveur
- Nous avons un ArrayList de GPS dans Activité
- Nous avons aussi l'ajout de deux méthodes dans Activité
    - addCoord(GPS obj)
    - startActivity()
    - stopActivity()

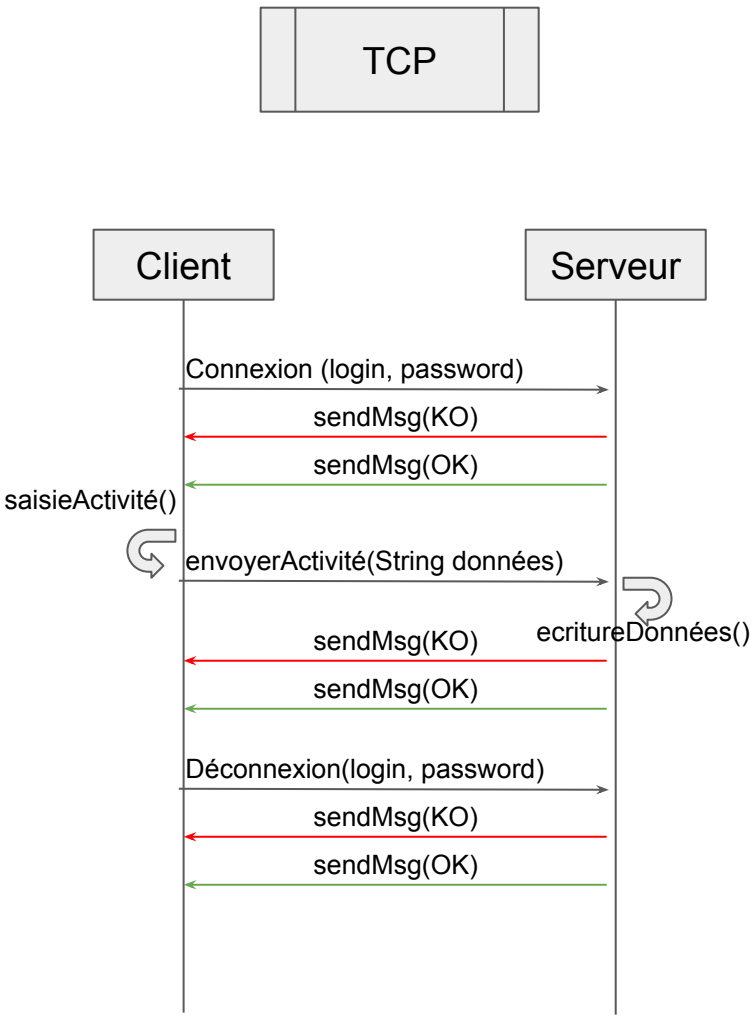
      Ces deux dernières méthodes servent à set les timestamp correctement

Pour l'implémentation multi-threadé en TCP, il faut aussi rajouter une classe ThreadConnexion.

De plus, contrairement à ce qui est indiqué dans le diagramme ci-dessus, les vérifications avec le GestionnaireUtilisateurs se font dans le Client, et non dans le Serveur.

Modélisations des échanges





Les chronogrammes ci-dessus sont de la théorie.
Nous ne les avons pas respecté à 100%, notamment pour tout ce qui est vérification des utilisateurs, qui se font dans le Client en pratique, et non dans le serveur.

Format JSON des messages

Nous n'envoyons qu'une seule donnée par message sous format JSON, ce sont les informations concernant l'activité.
Le message se présente comme suit :

Différence d'implémentation entre UDP et TCP

Globalement, UDP nécessite la création d'un DatagramPacket à chaque envoi de message, que ce soit côté Client ou Serveur. Envoyer un seul message ne pose pas de réelles complications, par contre, s'il s'agit d'une communication répétée entre le Client et le Serveur, le processus sera tout de suite plus lourd pour le programme.

TCP quant à lui permet une gestion rapide de l'envoi et de la réception de données. Il suffit simplement d'initialiser les flux d'entrées et de sorties en début de programme et les envois et réceptions se font de manière automatiques.

La modélisation client / serveur, contrairement à la modélisation par passage de messages, permet de retourner un message d'erreur ou de succès permettant de gérer correctement chaque cas d'utilisation. A l'inverse, la modélisation par passage de message ne permet pas d'être complètement sûr que les messages naviguent entre les deux entités. Bien qu'il puisse y avoir des messages contenant les erreurs, si le message se perd en route qu'il y ait une erreur ou non, l'autre entité ne recevra jamais aucun des deux.