04/2022 - 06/2022

Rapport ChatFusion

Un serveur de discussion TCP

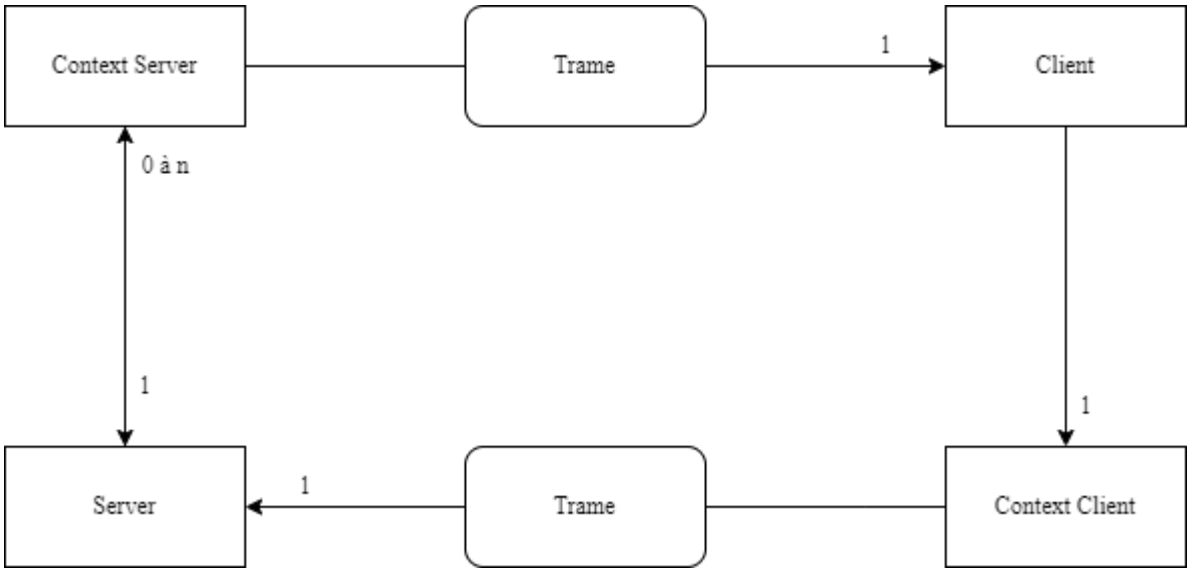
Leo Barroux
Robin Rieutord

Sommaire :

1. Architecture

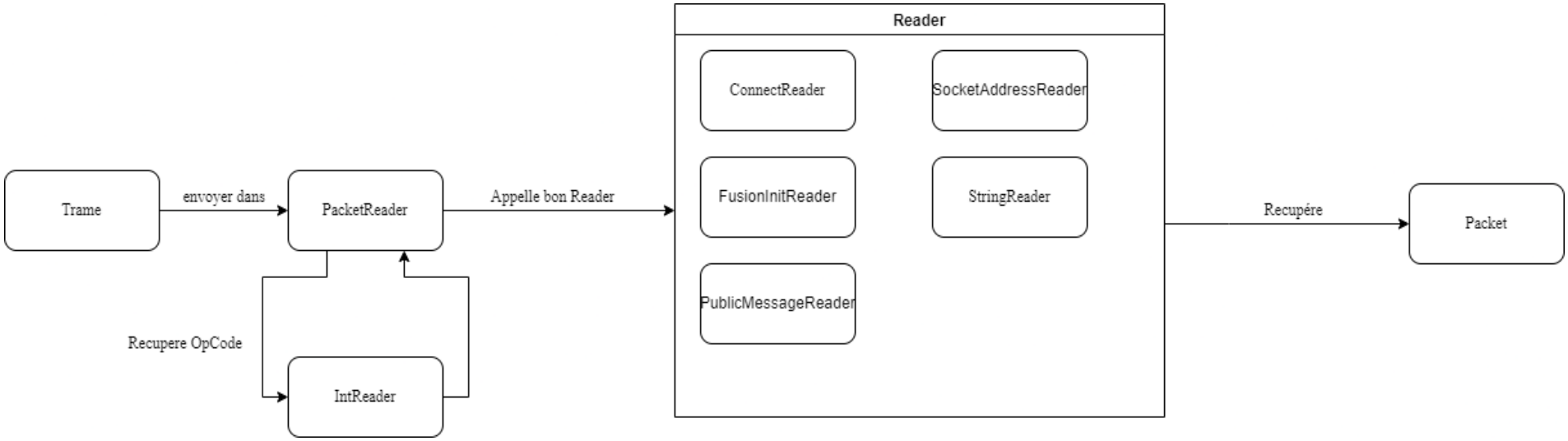
1.1 Global

L'application est divisée entre un serveur et un client, chacun est doté d'un Context qui servira à la communication. Un serveur et un client ou un serveur et un autre serveur communique par trames.
La lecture des trames s'effectue à l'aide de Reader et de packet.



1.2 Reader

Un reader permet de récupérer une trame d'un ByteBuffer et renvoie un packet. C'est le Paquet Reader qui va être appelé, récupérer la trame en appelant les différent Reader puis va renvoyer le bon Packet en fonction de l'OpCode.


Reader
ConnectReader
SocketAddressReader
FusionInitReader
StringReader
PublicMessageReader
Trame
envoyer dans
PacketReader
Appelle bon Reader
Recupere OpCode
IntReader
Recupére
Packet

1.3 Packet

Un paquet contient le contenu d'une trame de manière à pouvoir être exploiter plus facilement, elle contient son opCode, une méthode size qui permet de connaître la taille qu'elle prendra dans un buffer, une méthode components qui renvoie une liste des composants de la trame. Enfin un packet contient une méthode generateByteBuffer permettant de récupérer un buffer contenant la trame à partir des composants du packet. Cette méthode servira à construire des trames à envoyer grâce à la construction d'un packet. C'est d'ailleurs un packet qui est transmis dans les différentes méthodes depuis processIn jusqu'à processOut.

```
private void processOut() {
    var previewMsg :Packet = queue.peek();
    while (!queue.isEmpty() && bufferOut.remaining() >= previewMsg.size()) {
        var fullMsg :Packet = queue.poll();
        if (fullMsg == null) return;
        bufferOut.put(fullMsg.generateByteBuffer().flip());
    }
}
```

2. Etat d'avancement

2.1 Difficultés principales rencontrées

2.1.1 Construction de Reader

La première difficulté traversée fut sur la construction de reader, il était très difficile de récupérer le contenu des trams. Grâce au conseil reçu lors de la soutenance beta, nous sommes passés par un PacketReader et cela nous a aidé à voir nos erreurs.

2.2.2 Utilisation de SocketAddress

Il était difficile de comprendre ce qui devait être mis dans une trame SocketAddress et comment la récupérer d'une trame. Le SocketAddressReader à donc été une vraie difficultée. Dans l'autre sens, construire une trame avec une SocketAdress à partir d'un packet était compliqué. Grâce à une explication d'un de nos collègue, nous avons pu faire la lecture de la trame contenant un SocketAddress et pour créer une trame contenant une SocketAddress à l'aide d'un packet, nous avons réussi grâce à des recherche sur internet et à la javaDoc.

2.2 Evolution depuis la Beta

Toutes les sous-parties ci-dessous représentent une demande d'amélioration qui a été évoquée lors de la soutenance beta.

2.2.1 HashMap pour stocker les Clients

Initialement, notre serveur stockait tous ses clients dans une liste de clients, et ses serveurs connectés dans une HashMap de Server et de Context. Après discussion, nous sommes tombés d'accord sur le fait que stocker les clients dans une HashMap avec leurs Context respectifs serait plus efficace et plus pratique. Ci-dessous une capture d'écran montrant la création de la HashMap ainsi qu'une utilisation de celle-ci afin d'ajouter un client à la Map de clients connectés :

```
private final HashMap<Client, Context> connectedClients = new HashMap<>();
```

```
connectedClients.put(new Client(login), this);
```

2.2.2 Changer nom de méthode "parseToByteBuffer"

Le nom de la méthode "parseToByteBuffer" nous a également semblé incorrect, étant donné qu'il s'agit d'une méthode de l'interface Packet, qui renvoie un ByteBuffer en fonction des données du packet. Nous l'avons alors renommée "generateByteBuffer", comme nous pouvons le voir ci-dessous :

```
ByteBuffer generateByteBuffer();
```

2.2.3 PacketReader

Le fait d'appeler tous les Reader dans processIn dans un énorme switch n'aidant pas à la lisibilité, nous avons créer une classe PacketReader qui gérait qu'elle type de Reader il fallait appeler et qu'elle packet renvoyer. elle permet de récupérer un packet qui ensuite pourra être traité, en fonction de l'opCode qui y est stocké. C'est suite à ça que nous avons débloqué notre probléme de Reader.

```
Reader.ProcessStatus status = packetReader.process(bufferIn);
```

2.2.4 switch sur type de packet plutôt que opCode

Nous avons repris cette demande effectuée lors de la soutenance bêta, mais nous n'avons pas eu le temps de la mener à bien ni même pris le temps de réfléchir assez pour commencer à la développer, c'est donc l'une des demandes de la soutenance beta que nous n'avons pas pu mener à bien.

2.2.5 classe encapsulant comportement en fonction du Context

Idem que pour la partie précédente, cette fonctionnalité n'a pas eu le temps d'être menée à bien.

2.2.6 Changer les Packet de Class à Record

Comme les données des packet ne sont pas censées être modifiées ni accessibles en-dehors de la classe, nous avons par conseil du professeur remplacé tous nos Packet par des Record.

```
public record PacketFusionInit (int opCode,
                                String name,
                                SocketAddress sa,
                                1 usage
                                int nbMembers,
                                List<String> components)
        implements Packet {
```

```
public record PacketSocketAddress(int opCode, SocketAddress sa) implements Packet
```

3. Bilan fonctionnalités

En l'état actuel de notre programme, notre ChatFusion nous permet de créer des serveurs, ainsi que des clients. Les clients pourront se connecter à leur serveur indiqué en paramètre lors de la création. Une fois loggé (connecté) à leur serveur respectif, les clients pourront envoyer des messages publics sur le serveur afin que tous les autres clients puissent voir le message et y répondre. Nous avons ainsi un système de chat entre clients au sein d'un même serveur fonctionnel.

Il manque donc à ce projet l'envoie de message privé ainsi que de transfert de fichiers. Il manque également et surtout la partie fusion entre serveurs, qui a entièrement été développée au sein de notre projet mais qui possède toujours des erreurs que nous n'avons hélas pas réussi à régler à temps. Ces erreurs rendent ainsi l'utilisation de la fusion entre serveurs impossible.