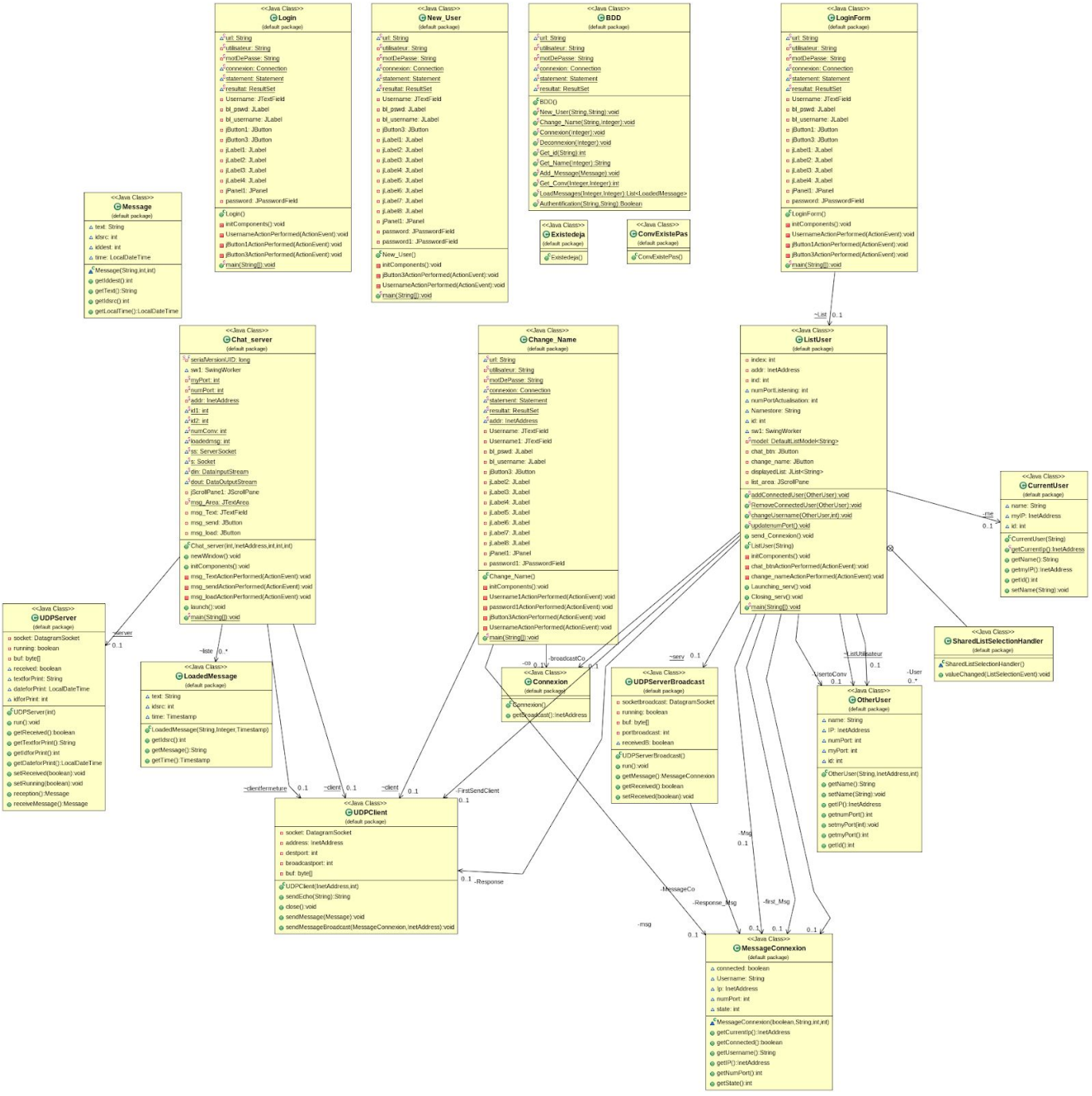
Rapport du Projet Chat System

Elie Périers - Quentin Larrede
4 IR - SI - A2

2019 - 2020, GEI INSA Toulouse

I SCHÉMAS DE CONCEPTION



voir le fichier image joint pour le détail

II IMPLÉMENTATION

Nous essaierons ici de décrire les principales étapes de l'implémentation, les difficultés que nous avons rencontré, ainsi que la façon dont l'implémentation concrète a pu affecter notre vision première du système.

Nous avons dès le départ choisi de fonctionner avec une stratégie d'implémentation progressive ; nous avons commencé par nous fixer des objectifs :

- sur le court terme (les quelques séances de TP à venir), très concrètes et divisibles en de petites tâches (broadcaster en UDP, sérialiser un objet…)
- sur le long terme, plus abstraits, comme "faire communiquer 2 utilisateurs à travers l'interface graphique", ou "gérer les identifiants des utilisateurs dans la base de donnée"

Au cours de notre progression, l'implémentation et la réflexion autour des objectifs à court terme nous a guidé pour la concrétisation des tâches sur le long terme ; l'orientation que nous avons choisi de prendre, par exemple, pour la partie réseau, par laquelle nous avons commencé, a grandement influencé l'organisation de l'interface graphique, par voie de conséquence.

Voici quelques exemples des lignes du fichier partagé qui nous servait de plan de développement, séance après séance :

- Envoyer un string en udp **BON**
- Envoyer un objet message en udp **BON**
- connexion : login + mot de passe : **BON**
- Envoyer un objet message à la BDD **BON**
- Broadcaster en UDP et recevoir **BON**
- Broadcaster un objet CONNEXION / DÉCONNEXION **BON**
- répondre à un objet CONNEXION avec son nom/adresse IP
- Envoyer une requête d'authentification à la BDD **BON**
- demander des messages dans une conv à la bdd **BON**
- Organiser la liste de conversation par plus récente
- Envoyer un message depuis la GUI **BON**
- Recevoir un message depuis la GUI

Les objectifs à long termes étaient ainsi naturellement remplacés par de petits objectifs ponctuels, réalisables facilement.

<u>PARTIE RÉSEAU</u>

Une grosse partie du début de notre travail a été de comprendre comment sérialiser et envoyer un objet en UDP. L'objectif était de pouvoir ouvrir un port, et d'y faire passer non pas un STRING, mais bien un objet JAVA, un "message", que nous avons conçu pour transporter toutes les informations pertinentes nécessaires lors de l'envoi d'un message entre 2 utilisateurs ; le texte, les ID des deux protagonistes, et la date TIME de l'envoi du

message (convertissable en format TIME SQL), pour pouvoir classer les messages dans la base de données.

Au tout début, pour tester nos fonctions sur le réseau du gei, nous avons décidé de différencier les ports pour gérer les types de messages à transmettre : le port 2050 à l'échange de messages entre les utilisateurs et le port 2045 servira aux broadcast pour lister les utilisateurs actifs. Il nous est rapidement apparu que cette méthode aurait ses limites, et que nous devions réfléchir à un système dynamique pour allouer les ports de connexions des différentes machines sur lesquelles notre programme sera installé. Pour cela, nous avons décidé de procéder comme suit :

Tous les messages qui ne sont pas directement destinés à la discussion entre deux utilisateurs sont transmis sur le serveur qui est toujours ouvert sur le port 2045 et qui attend des message broadcast. Ce socket là ne reçoit donc pas que les messages de broadcasts émis au moment de la connexion , mais tous les messages relatifs à l'état des utilisateurs (port d'écoute , connexion , lancement d'une session de clavardage, etc…).
Lorsqu'un nouvel utilisateur est découvert, un numéro de port va lui être attribué et sera le même pour tout le temps de sa connexion, il est ensuite communiqué, nous développons plus ce point dans la partie recherche des utilisateurs connectés.
Lorsque une session de clavardage est lancée , c'est à ce moment que le socket qui reçoit les messages contenant les informations échangés entre les utilisateurs se lance. Une fenêtre averti l'autre utilisateur que quelqu'un souhaite démarrer une session pour que lui aussi lance le socket.
A partir de ce moment , la communication est assez basique , c'est un échange d'objet message qui contient les informations tels que : le message à proprement parlé , les ids source et destinataire , l'horodatage du message etc…
Lorsque la session de clavardage est fermé d'un côté , elle l'est aussi de l'autre mais elle peut-être relancée à tout moment sur le même port.

BASE DE DONNÉE

Ensuite, nous nous sommes occupés de la base de donnée. Il a fallu récupérer le serveur dédié avec nos identifiants, créer la base ainsi que les tables dont nous avions besoin (voir partie conception), puis trouver un moyen de lier le code java avec la base de donnée. Nous avons donc créé une nouvelle classe contenant toutes les méthodes utiles à la communication avec la base de donnée, utilisant les drivers JAVA nécessaires et établissant une connexion avec la BDD quand nécessaire. La phase de connexion devant obligatoirement s'effectuer à l'intérieur d'un *try {},* il nous a fallu créer une connexion et la fermer à chaque appel de méthode. Les méthodes communicantes sont les suivantes :

- public static void New_User (String Name, String Password)

- public static void Change_Name(String NewName, Integer iduser)
- public static int Get_id(String Name)
- public static String Get_Name(Integer id)
- public static void Add_Message(Message Msg)
- public static List<LoadedMessage> LoadMessages (Integer idConv, Integer nb_msg)
- public static Boolean Authentification (String login, String pwd)

PHASE DE LOGIN

Nous avons décidé de créer une page de connexion classique (association login-mdp) pour identifier les utilisateurs. N'ayant pas de contraintes particulières de sécurité dans le cahier des charges, nous avons simplement utilisé une fonction de hachage pour stocker les mots de passe en SHA dans la base de donnée. L'algorithme de connexion vérifie alors que le login et les signatures des mots de passe correspondent bien avec ceux stockés dans la base de donnée. Une fois l'utilisateur connecté, la fenêtre de connexion se ferme, et le chat system s'ouvre, chargeant les utilisateurs connectés.

RECHERCHE DES UTILISATEURS CONNECTÉS

Deux méthodes distinctes permettent à un utilisateur qui se connecte d'avoir accès à la liste des utilisateurs connectés :

La première méthode, que l'on appellera "décentralisée", ne nécessite aucun échange avec la base de donnée. Lorsqu'un utilisateur se connecte, il envoie automatiquement un message de connexion en broadcast sur un port prévu à cet effet afin de prévenir les membres déjà connectés qu'il est présent en communiquant ses informations pour pouvoir être reconnus (username , id , adresse IP). Recevant ce message, les autres utilisateurs renvoient à leur tour un message de connexion en réservant et en communiquant le port sur lequel l'utilisateur nouvellement connecté peut le contacter. Quand le nouvel utilisateur a reçu les réponses à son message broadcasté il peut à son tour réserver et communiquer les ports d'écoute. C'est donc en 3 phase que la reconnaissance des utilisateurs connectés au réseau se fait .

La seconde méthode utilise la base de donnée. Dans la table des utilisateurs, nous avons rajouté un entier "available" qui est initialisé par défaut à "NULL". Celui-ci passe à 1 lorsqu'un utilisateur se connecte, puis passe à 0 dès lors que l'utilisateurs ferme la fenêtre de la liste des utilisateurs disponibles. Pour pouvoir utiliser le système de nouveau, il devra alors relancer une page de login, ce qui remettra l'entier "available" à 1, etc. Pour obtenir la liste des utilisateurs disponibles, il suffit donc de faire une requête SQL en triant les user avec le paramètre "available" à 1.

PARTIE SYNCHRONISATION

Pour satisfaire l'exigence [CdC-Bs-20] du cahier des charges, et de manière plus large les attentes de synchronisation de l'agent ditribué sur plusieurs machines, nous avons décidé de rajouter une catégorie spéciale dans l'objet "connexion". En effet, l'attribut "state" permet de discerner le type de message broadcasté. En l'occurrence, pour un changement de pseudo, l'attribut prend "5" pour indiquer au réseau qu'il s'agit d'un changement de pseudo. Chaque utilisateur connecté update automatique sa liste d'utilisateur connecté, en changeant simplement l'ancien nom par le nouveau, à l'emplacement correspondant à l'ID de l'utilisateur qui change de pseudo.

PARTIE GÉNÉRALE SUR LA GUI

Pour les différentes interfaces graphiques, nous avons majoritairement travaillé sur NetBeans. Nous avons construit les pages, ajouté les composants, puis transporté le code formé automatiquement vers Eclipse pour ajouter les fonctionnalités qui nous intéressaient. La page de connexion, de changement d'utilisateur et de création d'utilisateur ont été pensées à partir du même modèle dans un soucis de cohérence visuelle.

III TESTS & VALIDATION

A\ Tests unitaires

PARTIE BASE DE DONNÉES

Chacune des méthodes implémentées dans la classe qui gère la partie base de données ont été testée de manière indépendantes dans un main approprié, qui n'a servi que pour la partie développement. Pour chaque méthode, les cas critiques ont été testés ;

- changement de nom : nom vide, nom déjà existant, caractères spéciaux :
- ajout d'un message à la conversation : nombre important de messages ajoutés en un temps très court, chaînes longues, changement des id des users
- récupérer l'id à partir d'un nom : pseudo inexistant, mauvais types en entrée

Voici quelques-uns des tests effectués :

```
BDD.New_User("Flute", "Flute");
BDD.New_User("Grincheux", "Grincheuxx");
BDD.New_User("Simplet", "Simplet");
BDD.New_User("Myd", "Myd");



BDD.Change_Name("Jean Michel Tableau Veleda &àç_", 1);
BDD.Change_Name("Jean Michel Tableau Veleda &àç_", 22);


System.out.print(BDD.Get_id("Simplet \n"));
System.out.print(BDD.Get_Name(3));


BDD.Connexion(28);
BDD.Deconnexion(1);
```

Et le résultat dans la base de données :

```
+--------+-----------+-----------+------------------------------------------+
| iduser | name      | available | pswd                                     |
+--------+-----------+-----------+------------------------------------------+
|     28 | Flute     |         1 | f5854d682c2b07623fc9c8269fe67545cbf3b67b |
|     29 | Grincheux |      NULL | 22c47f703091587fd08e1a9f15dc99f56b6d3d55 |
|     30 | Simplet   |      NULL | e4d77cc62a7df5d54f06f799d8eb1962ad4b5361 |
|     31 | Myd       |      NULL | e7af75a02d36d64fbe850c26b12da5150acb47c6 |
+--------+-----------+-----------+------------------------------------------+
4 rows in set (0.00 sec)
```

Des tests sur la partie graphique ont aussi été effectués, pour vérifier que l'implémentation de la GUI était fonctionnelle avec les fonctions de communication avec la BDD implémentées. La page de login (LoginForm) a été testée avec des utilisateurs inscrits, des utilisateurs non valides, et des champs vides :

B\ Étapes de validation

Nous avons finalement mis au point 7 étapes simples de validation, reposant sur les fonctions précédemment citées et testées dans les cas extrêmes ;

1 - Installation de l'agent sur deux ordinateurs d'un même réseau :

2 - Lancement de l'agent et création d'un nouvel utilisateur sur chaque ordinateur (Bob et Jean-Michel Tabasco) :

3 - Échange de messages entre ces deux utilisateurs :

4 - Changement de pseudo d'un de ces deux utilisateurs :

5 - Bob met fin à la session d'échange :

6 - Bob relance la conversation et accède à un historique de messages :

7 - Les utilisateurs réduisent l'agent et le ferment (déconnexion effective):

IV ADMINISTRATOR GUIDE

Vous trouverez ici les informations utiles concernant l'utilisation et l'exploitation de notre logiciel. Nous avons classé par rubrique les cas d'utilisation et les conseils d'utilisation :

A - Installer l'agent

Pour installer l'agent, il vous suffit de copier le fichier .jar que nous vous livrons. Lancez ensuite un terminal dans le répertoire d'installation et changer les droits d'exécution avec "chmod +x Balzam.jar". Linux, windows et OSX permettent de changer le logo du programme avec le logo fourni.

B - Gérer la base de données

Pour vous connecter à la base de donnée, utilisez les commandes suivantes :

- mysql -h srv-bdens.insa-toulouse.fr -u tpservlet_02 -p
- entrez le mot de passe (qui est de toute façon en clair dans les fichiers du programme) : oovoot8H
- use tpservlet_02 ;

Puis gérez la base de donnée comme vous le souhaitez avec les commandes SQL usuelles dans les tables des utilisateurs (user) et des messages stockés (Conversation). 3 utilisateurs sont déjà créés, sans conversation entre eux.select

V USER GUIDE

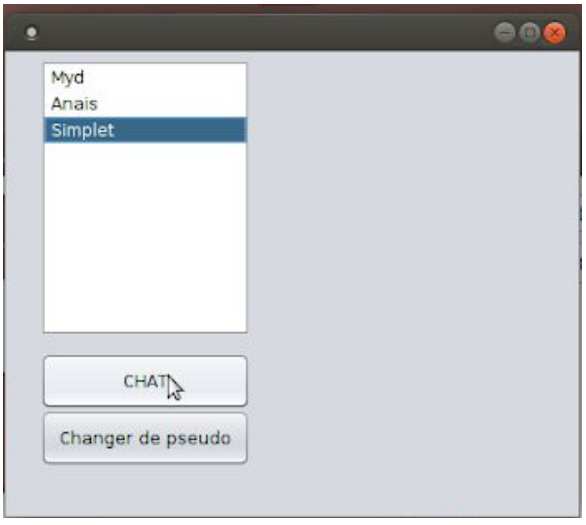
I/ Création de son compte utilisateur

Au lancement de l'application, une fenêtre s'ouvre, en bas de celle-ci, cliquer sur Nouvel utilisateur : une fenêtre s'ouvre. Vous pouvez alors simplement créer votre compte,
tout en sachant que votre nom d'utilisateur pourra être changé ultérieurement.

II/ Connexion

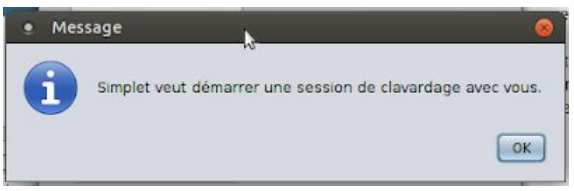
Après avoir créé son compte utilisateur, il suffit de rentrer son mot de passe et nom d'utilisateur au lancement de l'application.

III/ Démarrer une session de clavardage

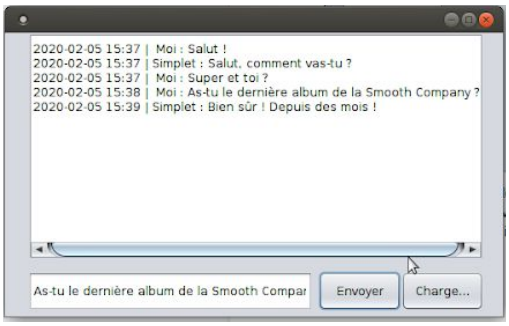


Pour démarrer une session de clavardage avec un utilisateur connecté, il suffit de cliquer sur le nom de l'utilisateur puis de cliquer sur le bouton CHAT. Cela ouvre directement la fenêtre de discussion, et l'utilisateur à qui vous voulez parler est lui-même notifié du début de la session de clavardage et la fenêtre de discussion est également lancé chez lui.

Attention : Il n'est pas possible d'ouvrir plusieurs fois une même conversation ! Il est par contre tout à fait possible d'ouvrir plusieurs fenêtre de conversation en même temps.



IV/ Pendant une session de clavardage



L'utilisation de la fenêtre de conversation est très simple, il suffit de cliquer sur le bouton envoyer pour envoyer votre message. Pour charger l'historique des messages échangé avec l'utilisateur , il faut cliquer sur le bouton Charge… , celà charge les 20 dernier messages envoyés. Si vous voulez les 100 derniers messages , il vous suffira de cliquer 5 fois sur le bouton.

Si la fenêtre pop-up ci dessous apparaît, c'est que votre interlocuteur à mis fin à la session de clavardage. Appuyer sur OK fermera la fenêtre de discussion. Cela ne veut pas dire que l'utilisateur n'est plus en ligne, vous pouvez relancer la conversation à tout moment.

Si vous voulez vous-même mettre fin à la session de clavardage , il vous suffit de fermer la fenêtre avec la croix en haut à droite de la fenêtre. Cela notifiera l'utilisateur avec lequel vous communiquiez. Il vous est également possible de réduire la fenêtre ou de l'agrandir avec les boutons usuels.

V/ Déconnexion de l'application

Pour fermer l'application et vous déconnecter , il suffit simplement de fermer cette fenêtre. Vous n'apparaîtrez plus dans la liste des utilisateurs des autres personnes connectées.

VII CONCLUSION

Le projet nous a demandé beaucoup de temps et de patience, pour plusieurs raisons. Premièrement, nous n'étions pas spécialement familiers du développement applicatif, Elie n'avait jamais écrit en java et Quentin n'avait suivi aucun cours de base de données. De plus, la partie réseau demande l'utilisation de fonction et librairies spécifiques, qu'il nous a parfois été difficile d'apprivoiser.
Nous avons tout de même réussi à réunir les compétences et le savoir nécessaire au développement d'une application qui remplit le cahier des charges. Le Balzam Chat System répond à ses attentes et permet, en plus, un système d'identification à clefs qui permet à un utilisateur de se connecter sur n'importe quelle machine sur laquelle l'agent a été installé.

Dans un cadre professionnel, il est clair que des améliorations seraient à prévoir ; la sécurité devrait être renforcée (notamment la gestion et les accès à la base de donnée), la partie graphique pourrait être améliorée et les "bonnes pratiques" de codage pourraient sûrement être plus strictement respectées. Malgré tout, notre projet est fonctionnel et répond au cahier des charges. Cela nous a permis d'acquérir des compétences techniques de manière efficace et autonome.

RESSOURCES UTILISÉES :

MySql :
https://dev.mysql.com/downloads/mysql/#downloads

Cours :
https://openclassrooms.com/fr/courses/626954-creez-votre-application-web-avec-java-ee/623973-introduction-a-mysql-et-jdbc

https://openclassrooms.com/fr/courses/626954-creez-votre-application-web-avec-java-ee/624392-communiquez-avec-votre-bdd

login/mdp :
https://openclassrooms.com/forum/sujet/authentification-java-mysql

https://www.dev2qa.com/how-to-use-mysql-on-mac/

driver .jar pour connexion database :
https://dev.mysql.com/downloads/connector/j/