
RAPPORT DE PROJET

Pikaviesti

Robin SIMON
Jean VAN INGHELANDT

Introduction

Ce projet a pour objectif principal la réalisation d'un système de clavardage distribué interactif multi-utilisateur temps réel permettant d'accroître l'efficacité des équipes d'une entreprise dans le respect d'un cahier des charges fourni. C'est dans ce but que nous, Robin SIMON et Jean VAN INGHELANDT, avons mis en place *Pikaviesti*, un logiciel de clavardage permettant aux intervenants d'échanger des messages textuels et des illustrations. *Pikaviesti* permet également la détection et la prise en compte des connexions des utilisateurs subites en cours d'exécution.

Nous avons deux mois pour réaliser ce projet que nous avons découpé grâce à quatre sprint sur l'outil Jira que nous avons déjà pris en main auparavant. Ce découpage nous a notamment permis de nous organiser efficacement en mettant en place un système de tâches et de priorité. En plus de cela nous avons également partagé le travail entre les membres de notre binôme. Ainsi, Robin SIMON s'est principalement penché sur le back-end¹ tandis que Jean VAN INGHELANDT a travaillé de son côté sur le front-end² en majorité.

Ce rapport se divise en plusieurs parties distinctes. Ainsi, vous verrez des étapes de la production de *Pikaviesti* telles que la conception et les différents choix de technologies que nous avons faits. Les différents tests que nous avons réalisés seront également passés en revue. Finalement, nous vous présenterons les étapes permettant la prise en main de *Pikaviesti*, en commençant par l'installation du système pour finir avec un manuel d'utilisation simplifié.

Ainsi, à la fin de la lecture de ce rapport, vous aurez pu découvrir comment nous avons réalisé cette application de la conception au développement. Vous saurez également comment installer et utiliser le système rapidement et simplement.

¹ Terme désignant un étage de sortie d'un logiciel devant produire un résultat.

² Partie d'un programme informatique responsable uniquement de l'interface utilisateur.

Conception et diagrammes réalisés

Lors de la conception de *Pikaviesti*, afin de ne pas nous lancer à l'aveugle dans la programmation de notre logiciel et donc d'optimiser notre travail, nous avons été amené à réaliser trois diagrammes de modélisation. En phase de production, les concepts et les solutions mises en place ont évolué de façon significative, ce qui nous a amené à mettre à jour les diagrammes autant que possible dans le temps imparti.

Les 3 diagrammes originaux sont les suivants : <https://imgur.com/a/BLXRf78>

La mise à jour du diagramme de cas d'utilisation n'a consisté qu'en l'ajout du cas d'utilisation "Exchange Image" qui utilise l'OS, l'utilisateur et le CommunicationSystem. Les fonctions Log In et Sign Up n'ont pas été ajoutées car l'agent utilisateur est un "ConnectedUser" ce qui implique qu'il est déjà connecté. Modifier le diagramme pour ajouter ces deux cas ne semble pas pertinent car cela complexifierait le diagramme inutilement, le log in étant un cas d'utilisation requis pour pouvoir exécuter tous les autres.

La mise à jour du diagramme de séquence n'a pas été faite car ce diagramme est devenu obsolète très rapidement. L'interface utilisateur s'est révélée être la source de la quasi-totalité des actions effectuées par le système, ce que le diagramme reflète très mal. Toutes les méthodes et objets manipulés sont à modifier. Seule la logique est intacte, quoique simplifiée.

Nous avons aussi ajouté le diagramme de classe généré par IntelliJ automatiquement, complet mais beaucoup moins clair.

Les mises à jour : <https://imgur.com/a/2CrhXMg>

Architecture du système et choix technologiques

Architecture du système

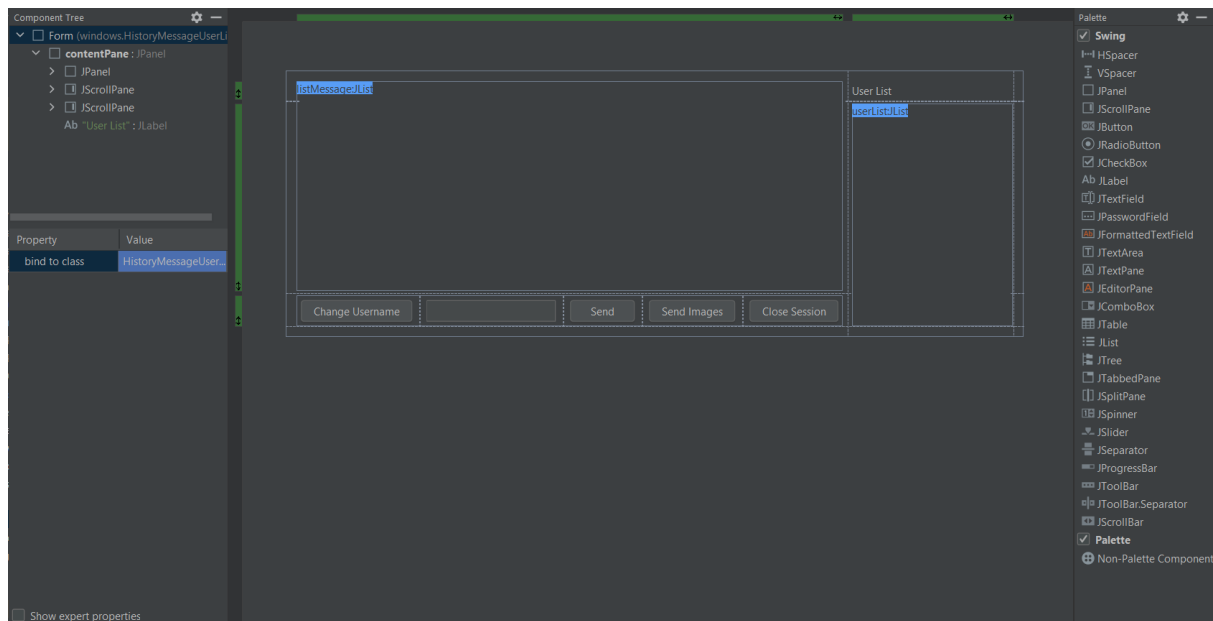
Le système a été créé avec le design-pattern MVC (Model-View-Controller) en tête (resp. ChatSystemModel.java, ChatSystemGUI.java et ChatSystemController.java). Le Model contient la liste des utilisateurs connectés ainsi que leur adresse. Le ChatSystemGUI ainsi que les diverses fenêtres associées gèrent l'affichage sur l'écran et ne gèrent donc que des String.

Pour simplifier le débogage et le codage de l'application, les messages envoyés suivent le format suivant : "id_source/id_destination/code_message/timestamp/content". L'id_source correspond à l'identifiant unique de l'utilisateur qui a envoyé le message et qui correspond à la clé primaire (nécessairement unique) du compte de cet utilisateur. Le code_message est ce qui permet de discriminer le type de message. Par exemple, 0 correspond à un message de chat qui doit (receiveMessage de CommunicationSystem) simplement être envoyé à l'interface graphique pour être affiché (voir Message.java pour la liste complète des correspondances). Le timestamp correspond à un objet LocalDateTime qui indique quand le message a été envoyé (contient l'heure, la minute et le jour). "Content" correspond au contenu du message. Les messages n'ayant pas de contenu (un broadcast pour connaître les utilisateurs connectés) ont un "content" = "fill" par convention. Il existe cependant un cas particulier : les messages de type 5 qui alertent le receveur que le prochain message est un ByteArrayInputStream à lire comme tel et qui ne présente donc pas les caractéristiques précédemment établies.

Choix technologiques

IntelliJ

Nous avons fait le choix d'utiliser IntelliJ afin de profiter de son UIDesigner qui permet d'obtenir une interface graphique claire et ce rapidement.



Cela a cependant empêché l'utilisation d'Eclipse car traduire un projet IntelliJ contenant UIDesigner produisait un code que Eclipse ne reconnaissait pas. Ces problèmes techniques nous ont cependant fait perdre un temps conséquent, tant en séance que hors séance. Il se pose alors la question de si cela en valait la peine, ce qui reste à débattre. Nous estimons que l'interface résultante est simple et ergonomique, et le code qui l'a produit est clair et concis. Nous ne regrettons donc pas notre choix même en prenant en compte les problèmes techniques qui peuvent être assez souvent reliés à notre inexpérience.

L'utilisation de IntelliJ a aussi permis de télécharger facilement les Drivers nécessaires (automatiquement) ainsi que d'avoir accès à une interface graphique permettant de visualiser la base de données, ce qui n'est pas essentiel mais reste pratique.

Nous avons utilisé Java 17 par défaut, c'était la version que nous avions à l'origine.

MySQL

La base de données est MySQL. Nous avons choisi d'utiliser une base de données centralisée afin de simplifier le problème. Cela permet de nous affranchir, par exemple, du problème d'authentification (s'inscrire sur une machine puis essayer de se connecter sur une autre) et du stockage des données (stocker toutes les données partout ou partiellement ?). Il semblait qu'une base centralisée serait bien plus simple à créer et à utiliser. Cette partie ayant été particulièrement rapide, nous pensons avoir fait le bon choix.

Elle ne compte que deux tables, une qui stocke les messages et l'autre qui stocke les utilisateurs. Les images ne s'affichent pas dans la conversation entre deux utilisateurs (elles s'ouvrent à côté) et ne sont donc pas stockées dans chat_history de la base de données qui n'est censée contenir que les messages à afficher dans la conversation.

GitHub

La majorité du projet s'est déroulée sur GitHub et sur la même branche (main). Nous n'avons créé de branche qu'en cas de dysfonctionnement global du projet qui nécessitait d'avoir une branche stable pour pouvoir continuer à avancer pendant le temps nécessaire pour trouver la source du problème ou dans le cas où une fonctionnalité majeure était en train d'être implémentée. Ces deux situations ne se sont produites qu'une fois chacune, la première correspondait à un push qui essayait de faire fonctionner un projet eclipse sur IntelliJ et la deuxième qui correspondait à l'élaboration du système d'envoi de message et de connexion TCP. Le travail ayant été bien réparti, nous n'avons pas eu de conflits pour push les différentes fonctionnalités.

La répartition des tâches a été faite très simplement : Jean s'est occupé de tout ce qui touche à l'interface graphique (fenêtres, pop up, design) et Robin s'est chargé de la logique (base de données, multithreading, communication). En bref, Jean était le développeur front-end et Robin le développeur back-end. Les seules classes qui ont demandé un travail en commun plus poussé sont les classes ChatSystemGUI et ChatSystemController car elles constituent le pont entre le front-end et le back-end.

Jira

Finalement, afin de nous organiser plus intelligemment, nous avons eu l'occasion d'utiliser Jira pour diviser les deux mois qui nous étaient accordés pour la réalisation du projet. Nous avons donc découpé le projet en quatre sprints puis en stories et enfin en tâches dont nous avons estimé la durée et la difficulté. Étant la première fois que nous utilisions cet outil après une brève introduction, nous avons plutôt mal estimé les durées nécessaires ce que nous avons tenté de rectifier tant bien que mal au début. Ces nouvelles estimations bien que plus proches de la réalité ne furent évidemment pas parfaites mais nous ont tout de même permis d'avancer sur notre travail efficacement.

Le premier sprint consistait principalement en la mise en place du système de création de pseudonyme que ce soit l'interface ou le principe de broadcast UDP. En effet, cette fonctionnalité était l'une de celles qui nous semblait la plus importante et à laquelle nous avons associé une priorité élevée. Durant le deuxième sprint nous nous étions fixé le but de mettre en place le principe d'envoi et de réception de messages notamment par la mise en place d'une interface claire et efficace et par l'utilisation de TCP pour le transfert. Finalement, les deux derniers sprints nous ont permis d'implémenter l'envoi et la réception d'images ainsi que le choix des sessions et le principe d'inscription à *Pikaviesti*.

Procédures d'évaluation et de tests

La majorité des tests ont été réalisés sur une même machine avec deux instances du projet *Pikaviesti*. Cet arrangement était amplement suffisant pour réaliser le débogage lors du développement. Des tests ont cependant été faits en conditions réelles afin de vérifier que le comportement de l'application sur deux machines différentes était identique au comportement sur une seule machine.

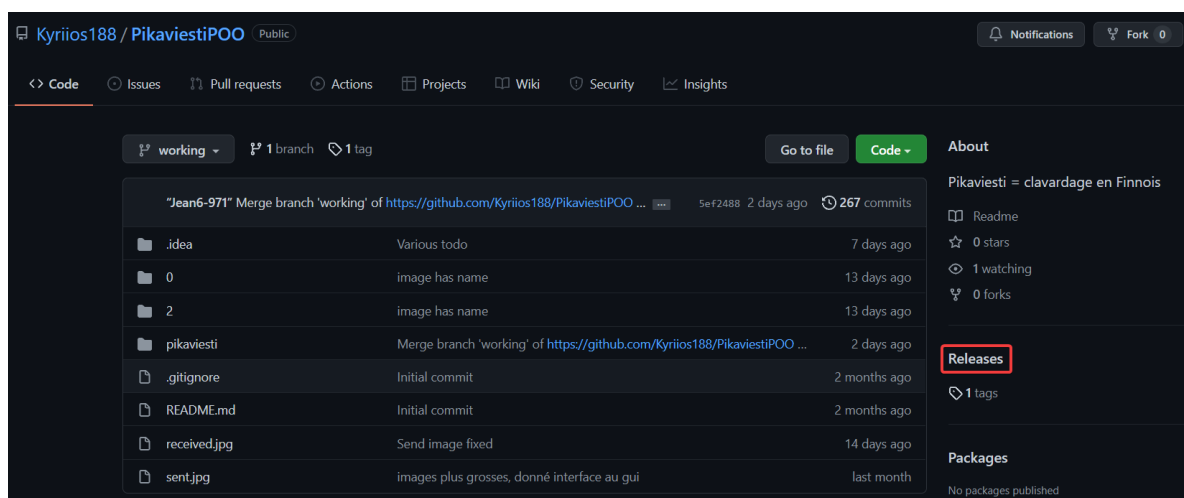
La totalité des tests se sont déroulés de la même façon : on établissait auparavant le comportement souhaité du programme face à une certaine séquence d'actions puis on vérifiait sur l'interface graphique si le programme réagissait d'une façon différente que prévue. On vérifiait ensuite, à l'aide de nombreux affichages sur la console, si le comportement réel était aussi conforme aux attentes. Cela permettait de séparer les problèmes dans deux catégories : les problèmes invisibles à l'utilisateur et les problèmes visibles à l'utilisateur. Par exemple, fermer le programme a longtemps créé des erreurs ou alors ne se fermait pas complètement. Ce problème était moins important que celui qui empêchait d'envoyer des messages après avoir envoyé une image (problème qui impact visiblement l'expérience de l'utilisateur).

Certaines méthodes de tests ont été écrites afin de ne pas avoir d'erreur dans la base de données. Les méthodes `loginExists` et `findAccount` permettent par exemple d'éviter d'entrer deux fois les mêmes informations dans la base, ce qui provoquerait une erreur. `createAccount` avertit l'utilisateur si des caractères spéciaux ont été utilisés alors que ce n'est pas autorisé et le choix de nom d'utilisateur aussi.

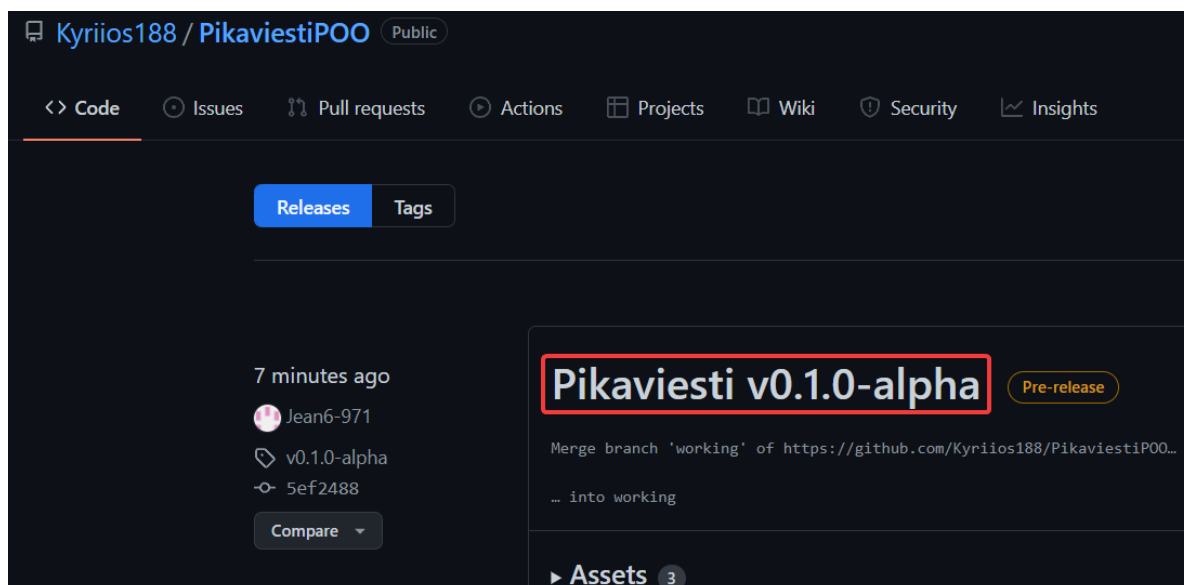
Procédure d'installation et de déploiement

L'installation et le déploiement de *Pikaviesti* sont simples, les étapes à suivre sont les suivantes :

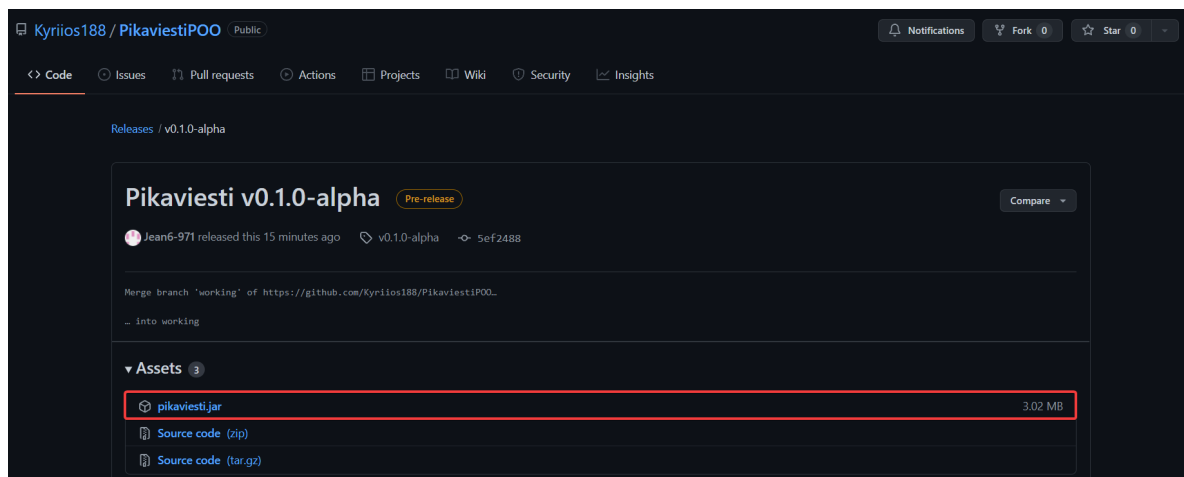
- vérifiez si java 17 est bien installé, si ce n'est pas le cas rendez vous sur le site <https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>, téléchargez la version qui vous convient le mieux et installez la
- allez sur notre dépôt Git : <https://github.com/Kyriios188/PikaviestiPOO>
- cliquez sur **Releases** (encadré en **rouge** ci-dessous)



- cliquez sur le nom de la dernière version disponible (nommé **Pikaviesti v0.1.0-alpha** dans le cas de notre exemple ci-dessous)



- téléchargez le fichier **pikaviesti.jar**



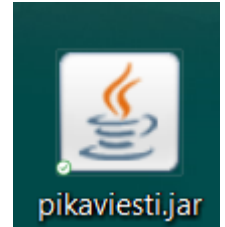
- enfin, déplacez le fichier vers l'emplacement qui vous plaît le plus comme votre bureau par exemple

Ainsi, *Pikaviesti* est installé, il ne vous reste plus qu'à suivre les indications du manuel d'utilisation simplifié pour découvrir comment utiliser l'application.

Manuel d'utilisation simplifié

Lancement

L'application se lance avec un double clique sur le fichier **pikaviesti.jar** :



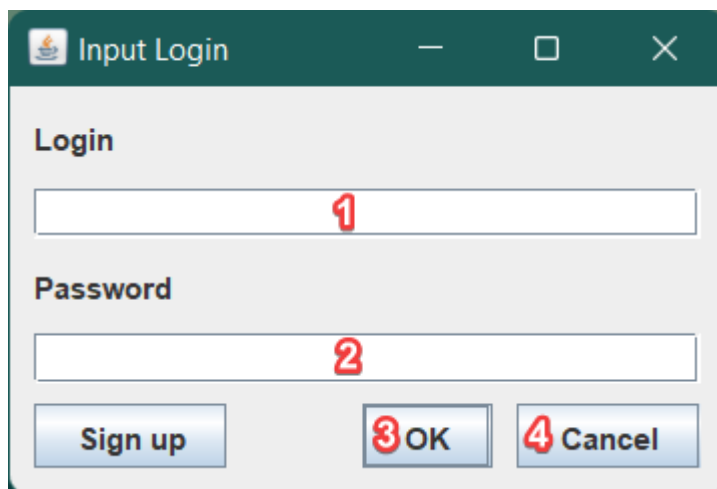
ou avec la **commande** :

```
java -jar ./pikaviesti.jar
```

Interface

Authentification

Dès lors, cette fenêtre apparaît :

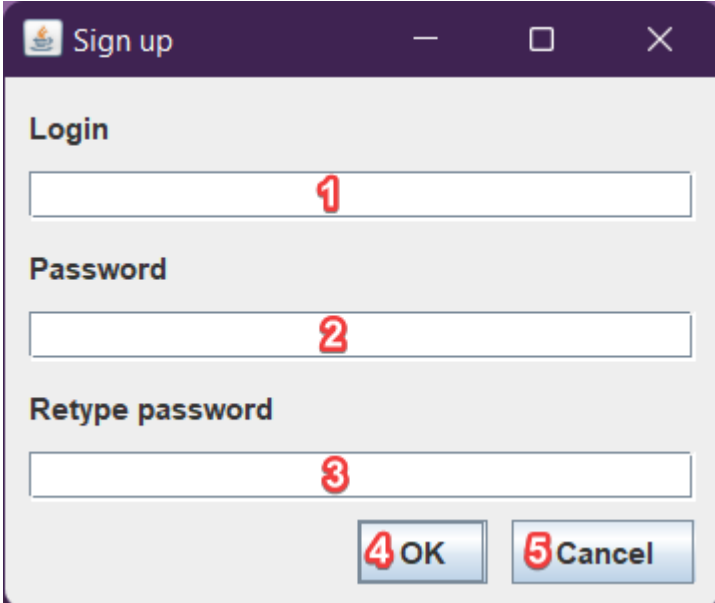


Elle permet l'**authentification** des utilisateurs. Pour cela, si vous avez déjà un compte, il vous suffit d'entrer son **nom d'utilisateur** dans la zone **1**, son **mot de passe** dans la zone **2** et de valider l'identification en cliquant sur le bouton **OK (3)**. Le bouton **Cancel (4)** permet de fermer le programme. Si vous n'avez pas encore de compte, vous pouvez cliquer sur le bouton **Sign up** en bas à gauche afin de vous inscrire.

Si les informations entrées dans les zones **1** et **2** sont correctes et que vous validez, la fenêtre de **choix du pseudonyme** s'ouvre.

Inscription

La fenêtre suivant apparaît dès l'appui sur le bouton **Sign up** de la fenêtre précédente :

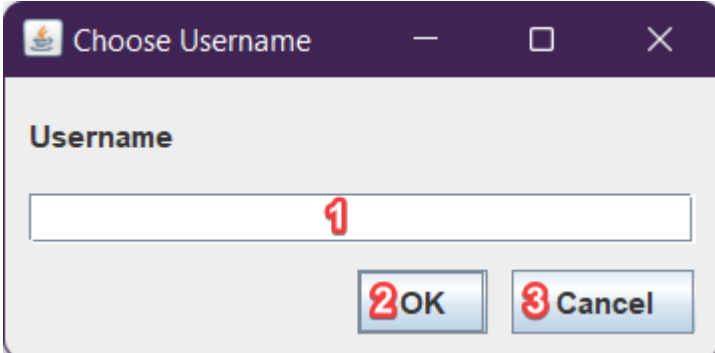
A screenshot of a 'Sign up' window. The window has a title bar with a small icon, the text 'Sign up', and standard window controls (minimize, maximize, close). The main area is light gray and contains three input fields. The first field is labeled 'Login' and has a red number '1' next to it. The second field is labeled 'Password' and has a red number '2' next to it. The third field is labeled 'Retype password' and has a red number '3' next to it. At the bottom right, there are two buttons: 'OK' with a red number '4' and 'Cancel' with a red number '5'.

Celle-ci permet l'**inscription** sur *Pikaviesti* en entrant son **nom d'utilisateur** dans la zone **1**, son **mot de passe** dans les zones **2** et **3** et en validant grâce au bouton **OK (4)**. Le bouton **Cancel (5)** permet de retourner sur la fenêtre d'**authentification**.

Si les informations entrées dans les zones **1**, **2** et **3** sont correctes et que vous validez, la fenêtre d'**authentification** s'ouvre à nouveau et vous permet de vous connecter au système.

Choix du pseudonyme

La fenêtre qui vous permet de choisir votre pseudo est la suivante :

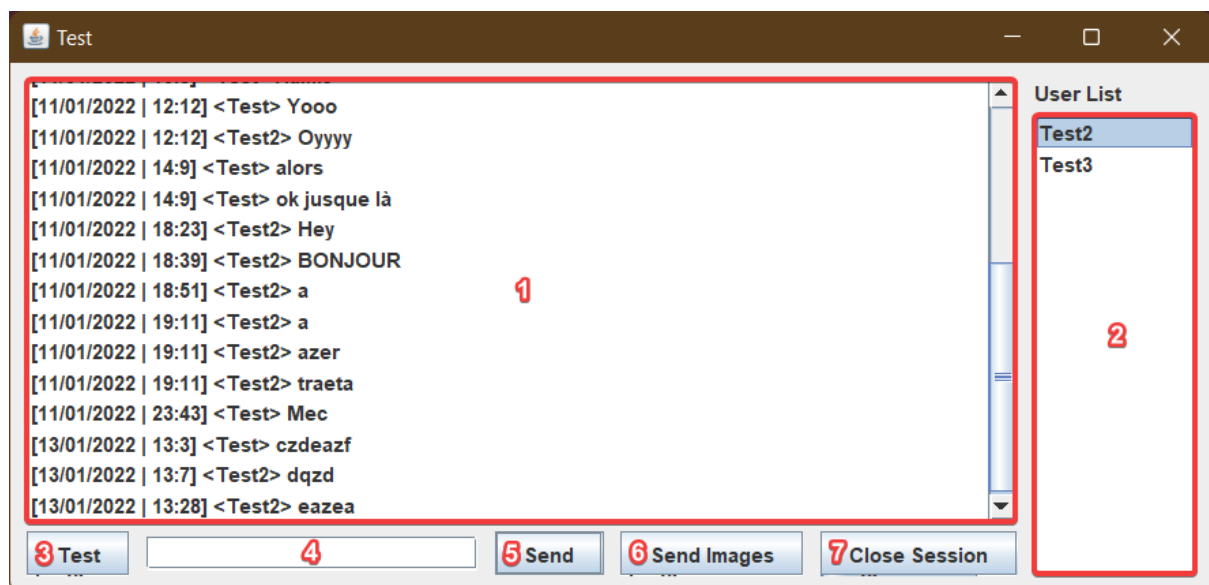
A screenshot of a 'Choose Username' window. The window has a title bar with a small icon, the text 'Choose Username', and standard window controls (minimize, maximize, close). The main area is light gray and contains a single input field labeled 'Username' with a red number '1' next to it. At the bottom right, there are two buttons: 'OK' with a red number '2' and 'Cancel' with a red number '3'.

Grâce à celle-ci, vous pouvez entrer le pseudonyme que vous choisissez dans la zone prévue à cet effet (1) et valider avec le bouton **OK** (2). Le bouton **Cancel** (3) ferme la fenêtre.

Si le pseudo entré dans la zone 1 est valide et si elle n'était pas déjà ouverte, la fenêtre **principale** de l'application s'ouvre. Un pseudo est considéré comme valide s'il est **unique** et ne contient ni '/', ni '\'.

Principale

La fenêtre principale comporte plusieurs zones distinctes ayant des fonctions diverses. Elle se présente sous la forme suivante :



La zone 1 contient l'historique des messages échangés avec la personne sélectionnée. Les messages sont divisés en quatre parties :

[11/01/2022 | 18:39] <Test2> BONJOUR

Ainsi, on retrouve la **date** et l'**heure** d'envoi du message, l'**auteur** et le **texte** envoyé.

La zone 2 contient quant à elle la liste des personnes connectées en même temps que vous. Un simple clique gauche sur le pseudonyme de la personne voulue et vous commencez une session de clavardage avec celle-ci, ainsi vous voyez les messages échangés précédemment avec cette personne dans la zone 1.

Le bouton 3 vous permet de changer de pseudonyme (sur l'exemple ci-dessus, le pseudonyme actuel est "Test") en ouvrant la fenêtre prévue à cet effet.

L'envoi de message se déroule de la manière suivante :

- choisissez un interlocuteur en cliquant dessus dans la zone **2**
- entrez le message voulu dans la zone **4**
- validez l'envoi du message en cliquant sur le bouton **Send (5)**

L'envoi d'images se déroule de la manière suivante :

- choisissez un interlocuteur en cliquant dessus dans la zone **2**
- ouvrez l'explorateur de fichiers en cliquant sur **Send Images (6)**
- choisissez l'image voulue et envoyez la en cliquant sur **Open**

Le bouton **Close Session (7)** permet de fermer une session actuellement ouverte avec la personne sélectionnée.