Mini-Chat

Responsable : Petru Valicov (valicov@lri.fr)

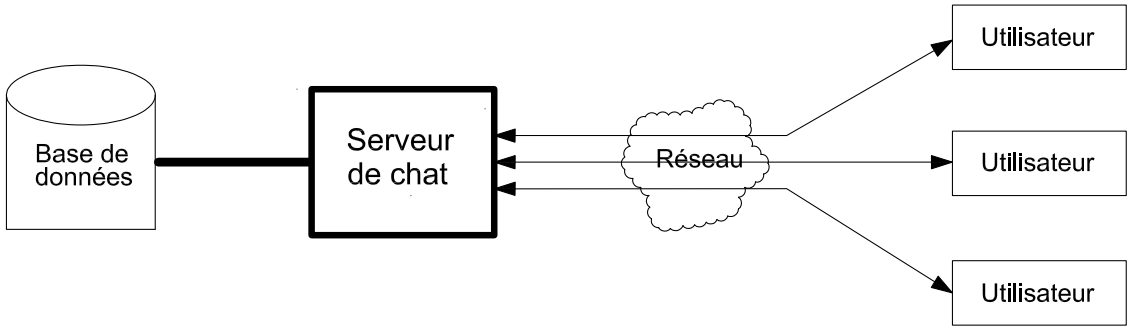
Le but de ce projet est de concevoir une application de messagerie instantanée aussi nommée Instant Messaging. Afin de réaliser ce travail vous serez amener à mettre en place les notions de génie logiciel, de modélisation, de programmation objets (vu le semestre dernier), les notions d'interface homme-machine, la programmation réseau (à travers l'interface des sockets) ainsi que les notions de bases de données avec Java.

Description générale

Le serveur mini-chat doit permettre à plusieurs utilisateurs de se connecter et de discuter entre eux à la manière des multiples logiciels de chat disponibles sur Internet.

Le principe de fonctionnement est le suivant : un utilisateur X écrit un message au serveur puis le serveur s'occupe de renvoyer ce message aux utilisateurs connectés auxquels X veut parler. Une fois connecté, l'utilisateur peut se déconnecter à tout moment. Au cours d'une session, un utilisateur peut envoyer certaines commandes particulières (à travers l'interface graphique) comme, notamment, une commande permettant de changer de pseudonyme ou récupérer la liste des utilisateurs connectés. Les utilisateurs utilisent un logiciel pour **se connecter au serveur** de chat et commencer à parler avec les autres utilisateurs. La possibilité de création d'un salon de discussion entre plusieurs utilisateurs doit être intégrée : un utilisateur crée un salon et d'autres utilisateurs peuvent s'y connecter (dans ce cas les messages envoyés seront diffusés à tous les membres du salon). Chaque utilisateur a la possibilité de consulter l'historique de ses discussions stocké dans une base de données. Pour rendre toutes ces fonctionnalités "user friendly", le logiciel a une interface graphique (IHM). Elle doit permettre aux utilisateurs de définir leurs statuts, lister les membres connectés et leurs attributs (pseudo, statut, etc.) et de discuter avec eux.

Le schéma ci-dessous présente la vision globale du système informatique demandé.



Pour se fixer un cadre de travail, le nouveau système informatique doit respecter certaines contraintes de compatibilités du client qui a commandé son développement (à l'occurrence il s'agit de l'enseignant). Ainsi, les logiciels seront obligatoirement développés en Java (version 1.6 ou plus) et devront fonctionner sous Linux et Windows. Le réseau sera obligatoirement un réseau IPv4 (TCP et/ou UDP).

Le projet est à faire en binôme ou en trinôme (sachant que le travail en trinôme prévoit une charge de travail plus élevée).

Fonctionnalités additionnelles

La description générale ci-dessus correspond aux fonctionnalités du système informatique basique. On aimerait que le système ait plus de fonctionnalités que cela, afin de le rendre plus original et attractif. Choisissez dans la liste ci-dessous au minimum une fonctionnalité additionnelle (pour les trinômes il faut en choisir au moins 2).

- Tri intelligent des contacts : pour chaque utilisateur la liste des autres utilisateurs est triée en fonction de ses préférences (la dernière date de discussion, fréquence des discussions, le nombre de messages échangés, le statut). Il s'agit d'obtenir un ordre aussi pertinent que possible !
- Cercles d'amis et blocage : chaque utilisateur veut pouvoir créer son cercle d'amis et/ou bloquer certains utilisateurs. Ainsi, X peut parler à Y uniquement si X est l'*ami* de Y et que Y n'a pas *bloqué* X. Il faudrait penser à reconfigurer la liste des utilisateurs pour chacun d'entre eux en fonction de ses amis.
- Robustesse : plusieurs serveurs (chacun avec sa propre base de données) se synchronisent entre eux. Si un des serveurs devient inaccessible, les autres prennent le relais.
- Interface graphique avancée : possibilité d'utiliser différentes polices de caractères, des couleurs, gestion des avatars

Les fonctionnalités proposées sont modifiables en fonction de vos goûts mais vous devez attendre que vos changements soient approuvés par le client avant de les intégrer dans votre projet. Par exemple, si vous implémentez les cercles d'amis, vous pouvez décider de faire en sorte qu'ils soient asymétriques (Y peut avoir X dans son cercle d'amis même si X a bloqué Y). Mais les détails de cette approche devrait être expliqués et justifiés auprès du client. Il en est de même si vous proposez des suggestions d'autres fonctionnalités supplémentaires (non triviales) - ne vous lancez pas dans l'implémentation sans l'approbation du client.

Rendus et soutenance

Vous devez rendre les différents documents et le code source au fur et à mesure de l'avancement de votre projet. Cela devrait vous permettre de se fixer les délais de mise en place de chaque étape de votre projet.

1. *Architecture du système.* **Remise : jeudi 7.02.2013 ; Présentation : lundi 11.02.2013**
   Il s'agit d'un document de 5 pages environ. Il contient obligatoirement les diagrammes UML suivantes :
   (a) diagramme de cas d'utilisation
   (b) diagramme de classe contenant au moins toutes les classes métiers (les classes métiers sont celles qui représentent des concepts du monde réel, les classes non métiers servent uniquement à des tâches informatiques (communiquer avec la base de données, créer l'interface graphique,...)).
   (c) diagramme de séquences

   Également, il doit contenir une description textuelle des classes importantes (un ou deux paragraphes par classe). Enfin, le document contient obligatoirement une description des algorithmes importants (trois ou quatre paragraphes par algorithme, avec données d'entrée, données de sortie, traitements).
2. *Manuel utilisateur.* **Remise : lundi 11.03.2012**
   Un document à destination des différents utilisateurs de votre système (les utilisateurs de base, mais aussi l'administrateur du serveur par exemple). Il doit indiquer comment

installer le ou les logiciels développés, comment les configurer et les lancer (par exemple, liste des arguments en ligne de commande), et comment les utiliser (liste de toutes les fonctions, scénarios d'utilisation,...). Des captures d'écran peuvent être incluses, mais elles ne remplacent pas un texte clair et précis. Le manuel utilisateur doit être le plus court possible tout en étant absolument complet. Les documents techniques (rendu 11/03) doivent permettre aux informaticiens du client de prendre en charge la maintenance et l'évolution de votre système. Ils peuvent ré-inclure les informations d'architecture du système si celles-ci ont changées depuis le 8/02 (ce qui, au passage, est un signe de mauvaise gestion du projet).

3. *Document technique.* **Remise : mercredi 13.03.2012**
   Les documents techniques sont destinés aux informaticiens et ils doivent contenir une description complète de toutes les parties du système informatique. Par exemple : description de chaque classe/package ou inclusion d'une javadoc détaillée, schéma des tables dans la base de données, description du protocole de communication entre le serveur de chat et les logiciels des utilisateurs, comment recompiler le projet et régénérer les .jar, librairies externes utilisées, jeux de tests, etc.
4. *Code source et exécutables.* **Remise : lundi 18.03.2012**
   Le code source devra être une archive `.zip` ou `.tar` contenant l'ensemble du code source **commenté** de votre projet : tous les fichiers `.java`, et les fichiers de ressources (`.jar` externes), fichiers d'images, fichiers `.sql` pour créer les tables, fichiers de configuration d'exemple, fichiers de projet Eclipse pour recompiler et tout ce qu'un un informaticien a besoin pour pouvoir utiliser vos logiciels. Vous devriez tester, sur un ordinateur séparé qui n'a jamais été utilisé pour le projet, que vous arrivez à recréer et exécuter les exécutables de cette archive `.zip`. Les exécutables sont les fichiers `.jar` contenant le code compilé (`.class`), un manifest exécutable, et les ressources du projet. Aussi, vous devriez effectuer la phase de tests de votre logiciel sur une machine autre qui n'a jamais été utilisée pour le projet. Il y a au minimum un fichier `.jar` par exécutable (il y en a plus si vous utilisez des librairies, soit externes, soit que vous avez créées).

Soutenance

Date : dans la semaine du 18.03 au 22.03 (à définir)

La présentation finale prend la forme d'une démonstration scénarisée de l'ensemble du système. Chaque groupe doit prévoir un scénario intégrant le maximum de fonctionnalités de l'application. Les membres du groupe peuvent jouer le rôle des différents utilisateurs du système. Plusieurs ordinateurs peuvent être utilisés si nécessaire. Le vidéo projecteur peut être utilisé pour projeter l'image de l'application en train d'être utilisé, ou pour projeter des slides, ou les deux en alternance. Le but principal est de montrer que votre système informatique est simple à utiliser tout en étant complet, que l'interface est ergonomique mais qui donne une certaine flexibilité à l'utilisateur, que le système est fiable et évolutif, etc. Vous auriez que 10 minutes (+ 5 minutes de questions) pour votre présentation. Les présentations intermédiaires sont plus informelles, sous la forme d'une petite réunion de 5 à 10 minutes.

Remarques d'ordre général

Les dates de remise doivent être respectées. Tous les rendus se font par e-mail à l'adresse `valicov@lri.fr`, le jour indiqué **avant minuit**. Aucun retard ne sera accepté.

Tous les documents doivent être impérativement en format `.pdf`. Même si ce n'est pas obligatoire, ils vous est vivement conseillé d'utiliser LaTeX. Sa maîtrise vous sera utile car c'est un

excellent outil pour la production scientifique.

Il est **strictement interdit** de copier du code du projet des vos collègues. Attention : l'obfuscation du code n'est pas une solution, c'est très facile à détecter ! En revanche, **vous pouvez** réutiliser des librairies externes existantes, mais vous devez l'indiquer et surtout expliquer comment vous les avez adaptés dans votre logiciel.

Préparez vos questions pour chaque séance.