INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

RAPPORT DE PROJET

Projet de COO-POO de 4IR : Application de clavardage

CAVAILLES Kevin, GUES Marianne

4IR-A

Table des matières

Installation et administration

***Note** : L'application dans son état actuel fonctionne uniquement en local, sur une seule machine. De plus, nous n'avons prévu que quatre paires de ports pour les entités de communication (UDP/TCP). Ainsi, seulement quatre applications peuvent tourner en même temps sur la même machine. Nous allons donc présenter l'installation et l'administration de l'application en local mais nous expliquerons néanmoins les différences et ce qu'il faudrait faire pour une installation sur plusieurs machines.*

Commencez par récupérer les exécutables ou source du projet.

Prérequis : Installer Git.

Sous Windows, lancez un git bash.

Exécutez la commande suivante dans le dossier de votre choix :
« git clone https://git.etud.insa-toulouse.fr/kcavaill/Projet_COO_POO ».
Cela doit faire apparaître trois dossiers et un fichier : « application/, modélisation/, serveur_presence/ et readme.txt ».
Il s'agit des fichiers présents sur la branche « master » du projet. Si vous voulez accéder aux sources plutôt qu'aux fichiers exécutables, passez sur la branche « application » en exécutant la commande : « git checkout application ».
Dans la suite, nous considérons que vous êtes sur la branche « master »

Dans le dossier application se trouve deux fichiers .jar exécutables et un dossier qui contient les librairies nécessaire à l'application.

chat.jar :
Exécutable du système de clavardage interactif. Le lancer une première fois permet de créer, s'ils n'existent pas, le fichier « database0.db » et le dossier « downloads/». Le premier est le fichier utilisé par l'application pour gérer les authentifications et l'historique des messages. Le second est un dossier dans lequel seront stockés tous les fichiers envoyés par d'autres utilisateurs au cours des sessions de clavardage.

create_users.jar :
Exécutable permettant d'insérer de nouveaux utilisateurs (de manière chiffrée. Voir Choix techniques et implémentation) dans la base de données de l'application à partir d'un fichier texte contenant des paires « nom_utilisateur :mot_de_passe ». Le lancer une première fois permet de créer, s'ils n'existent pas, les fichiers « database0.db » et « credentials.txt ».
Le premier est le fichier utilisé par l'application pour gérer les authentifications et l'historique des messages. Dans ce cas précis, le fichier est utilisé pour insérer de nouveaux utilisateurs dans la base de données. Le second est un fichier texte contenant des instructions quant au format requis pour pouvoir insérer de nouveaux utilisateurs.

Installation, démarche conseillée

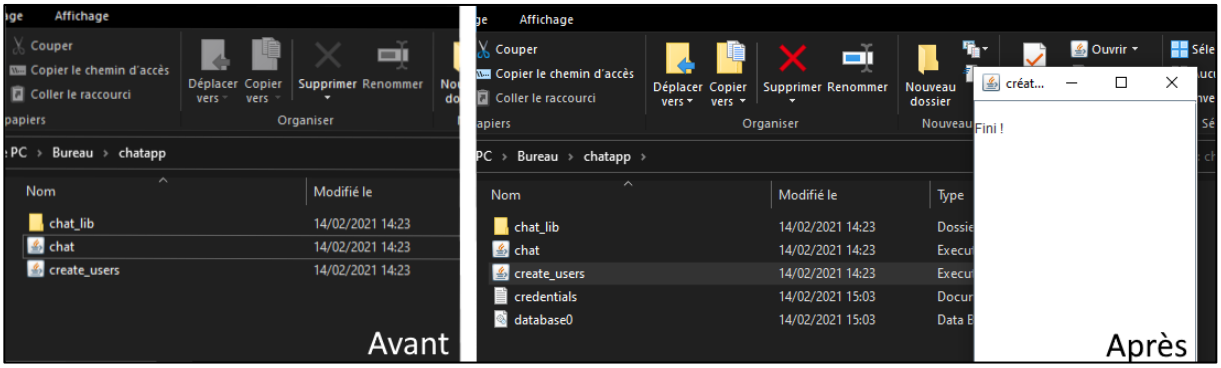
***Note** : Pour exécuter un .jar, vous pouvez cliquer dessus à l'aide de l'interface graphique ou bien, dans une console, vous placer dans le dossier du fichier et exécuter la commande : java -jar [nom_fichier_jar] ([arg1] [arg2]..)*

Prérequis : Installer le JDK-13.0.2 trouvable à cette adresse : https://jdk.java.net/archive/
et changer les variables d'environnement si l'installation ne le fait pas automatiquement :
https://superuser.com/questions/1057532/how-to-change-java-version-on-windows-10.
Vous pouvez vérifier quelle version est utilisée par votre machine en exécutant la commande «java -version » dans une console.

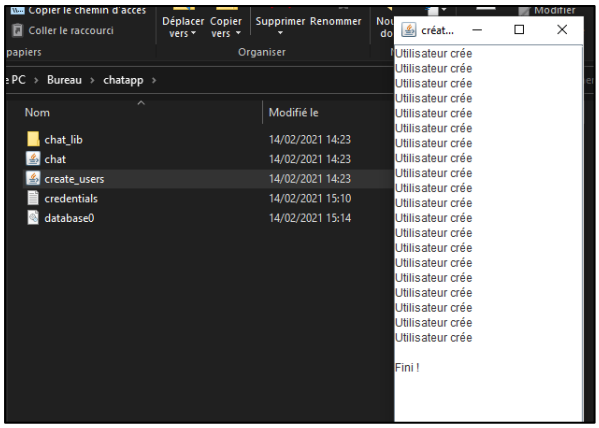
Pour l'application principale

- En local :

  Lancer une première fois « create_users.jar » afin de créer le fichier de base de données et le fichier texte avec les instructions pour créer de nouveaux utilisateurs :



  Ouvrir le fichier « credentials.txt », lire les instructions et insérer une liste de paires « nom_utilisateur :mot_de_passe », avec une paire par ligne. Chaque ligne vide, s'il y en a, doit être commentée en commençant par « // ».
  Eviter de revenir à la ligne à la fin du fichier.
  Relancer « create_users.jar » afin de créer les utilisateurs :



  Vous pouvez dès à présent tester l'application (Voir Guide utilisateur).

- Sur plusieurs machines :
  Il existe deux façons de procéder en fonction de la manière dont on souhaite administrer l'application :

  - Récupérer le projet sur une session/machine appartenant à un administrateur.
    Suivre les mêmes instructions que pour l'installation en local. Une fois les utilisateurs insérés, copier tous les fichiers et dossiers sauf « create_users.jar » et « credentials.txt » dans un nouveau dossier. Placer ce dernier sur les machines destinées à utiliser l'application (En le collant depuis une clé usb/disque externe ou en le téléchargeant depuis un serveur).
    Tous les utilisateurs disposeront ainsi de la même base de données au départ et tous pourront se connecter depuis n'importe quelle machine possédant l'application.

  - Récupérer le projet sur chacune des machines destinées à utiliser l'application. Créer un seul utilisateur par machine. Les bases de données seront indépendantes dans le sens où un utilisateur ne pourra se connecter que sur la machine qui possède ses données.

Pour le serveur de présence

Ouvrez le dossier serveur_presence/, chargez le fichier serveurpresence.war dans votre serveur tomcat et exécutez-le. Le dossier serveur_presence/ contient également un fichier application.jar, contenant une version modifiée de l'application de clavardage. N'utilisez pas la version classique de l'application avec le serveur, elles ne sont pas compatibles. La version modifiée fonctionne exactement de la même manière que l'application classique.

Vous pouvez également récupérer le code source sur la branche serveur_presence du dépôt git (dossier POO_Server), compiler le projet et exécuter le fichier src/main/ServletPresence.java. La version modifiée de l'application se situe dans le dossier POO ; compilez-le et exécutez le fichier src/main/Main.java.

Administration

Il est fortement conseillé d'installer SQLiteStudio, trouvable à cette adresse : *https://sqlitestudio.pl/*,
afin d'effectuer des opérations de modification et suppression qui ne sont pas possibles autrement (outils non développés).

- En local :
  - Ajouter de nouveaux utilisateurs :
    Utiliser « create_users.jar » pour ajouter des utilisateurs afin qu'ils puissent être authentifiés correctement (Certaines données sont en effet chiffrées lors de la création d'un utilisateur. Voir Choix techniques et implémentation).
  - Modifier des utilisateurs existants :
    Utiliser SQLiteStudio. Ajouter le fichier de base de données. Manuellement ou à l'aide d'une requête, modifier les noms d'utilisateurs. Le changement du mot de passe n'est pas supporté pour le moment.
  - Supprimer des utilisateurs existants :
    Utiliser SQLiteStudio. Ajouter le fichier de base de données. Manuellement ou à l'aide d'une requête, supprimer les utilisateurs, les conversations et les messages qui leurs sont rattachés.

- Sur plusieurs machines :
  - Ajouter de nouveaux utilisateurs :
    Utiliser « create_users.jar » pour ajouter des utilisateurs afin qu'ils puissent être authentifiés correctement (Certaines données sont en effet chiffrées lors de la création d'un utilisateur. Voir Choix techniques et implémentation).
    Deux choix possibles :
    1. Ajouter un nouvel utilisateur uniquement sur la machine qui lui est destinée. Il ne pourra pas s'authentifier sur d'autres machines.
    2. Ajouter un nouvel utilisateur sur une base de données existante (celle que l'administrateur crée au départ). Transmettre à toutes les machines possédant l'application la nouvelle base de données + un outil (à développer) permettant de remplacer uniquement la table des utilisateurs de l'ancienne base par la nouvelle.
  - Modifier des utilisateurs existants :
    Utiliser SQLiteStudio. Ajouter le fichier de base de données. Manuellement ou à l'aide d'une requête, modifier les noms d'utilisateurs. Le changement du mot de passe n'est pas supporté pour le moment.
    Transmettre à toutes les machines possédant l'application la nouvelle base de données + un outil (à développer) permettant de remplacer uniquement la table des utilisateurs de l'ancienne base par la nouvelle.

  - Supprimer des utilisateurs existants :
    Utiliser SQLiteStudio. Ajouter le fichier de base de données. Manuellement ou à l'aide d'une requête, supprimer les utilisateurs, les conversations et les messages qui leurs sont rattachés.
    Deux choix possibles :
    - Si les bases de données sont indépendantes, ne supprimer l'utilisateur que de la base de données de la machine qu'il utilise.
    - Sinon, transmettre à toutes les machines possédant l'application la nouvelle base de données + un outil (à développer) permettant de remplacer uniquement la table des utilisateurs de l'ancienne base par la nouvelle.

Guide utilisateur

Lancement de l'application

Il s'agit de la seule étape qui diffère réellement entre une utilisation locale et une utilisation sur plusieurs machines.

- En local :
  Afin de tester l'application localement, vous devez lancer plusieurs instances sur la même machine. Chaque instance requiert des ports pour les entités de communication. Ainsi, afin de pouvoir attribuer différents ports à chaque instance, il est nécessaire, dans une console, d'ajouter un argument à la commande d'exécution du .jar de l'application :

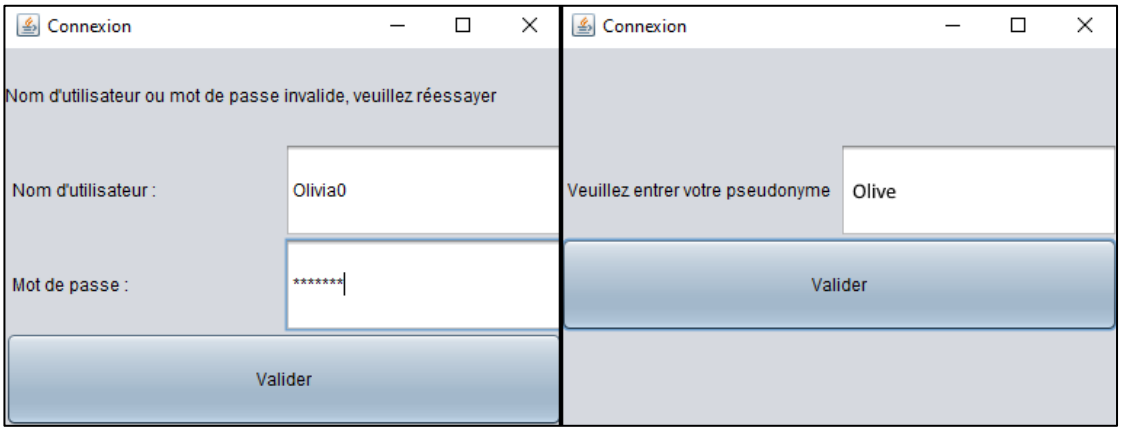
  « java -jar chat.jar X » avec X entre 0 et 3.

- Sur plusieurs machines :
  Si vous deviez tester l'application sur un réseau local, vous devriez lancer une seule instance de l'application sur plusieurs machines. Les entités de communication auraient alors les mêmes ports, ce qui ne poserait pas de problème. La commande à exécuter serait alors la même que précédemment mais sans argument :

  « java -jar chat.jar »

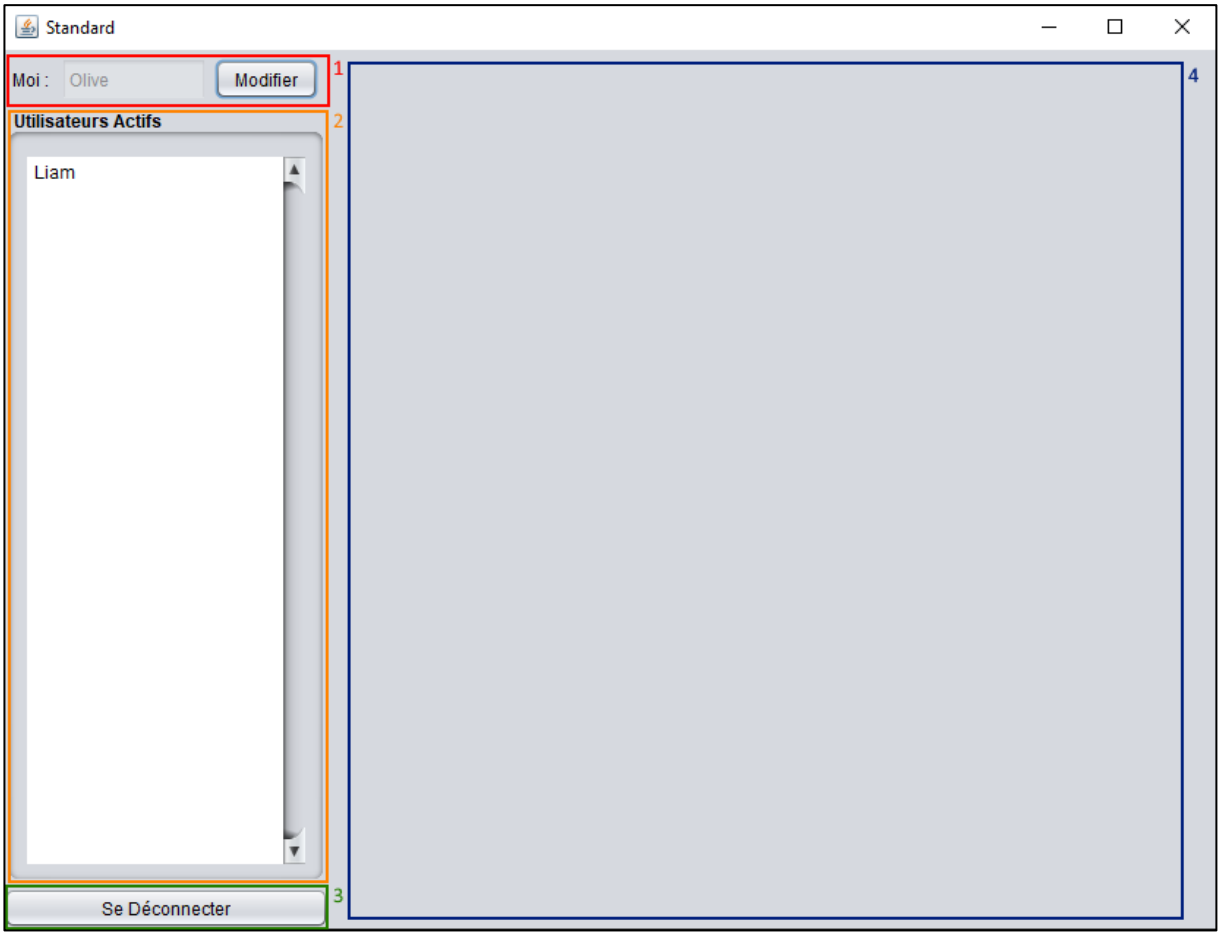
Connexion

Une fois l'application lancée, vous devez vous authentifier avec votre nom d'utilisateur unique et le mot de passe fournit par l'administrateur du système. Si vos identifiants sont incorrects, le message « Nom d'utilisateur ou mot de passe invalide, veuillez réessayer » s'affiche.
Une fois connecté, vous devez choisir un pseudonyme. Celui-ci doit faire au moins un caractère et ne doit pas déjà être utilisé par un autre utilisateur de l'application. Pour éviter toute confusion, vous ne pouvez pas choisir un pseudonyme déjà utilisé avec une casse différente. Exemple : Si « Alex » est déjà choisi, vous ne pouvez pas choisir « alex » comme pseudonyme. Si le pseudonyme que vous choisissez est déjà pris, le message « Ce nom est déjà utilisé, veuillez en choisir un autre » s'affiche.



Fenêtre principale

Après avoir choisi un pseudonyme, vous accédez à la fenêtre principale de l'application :



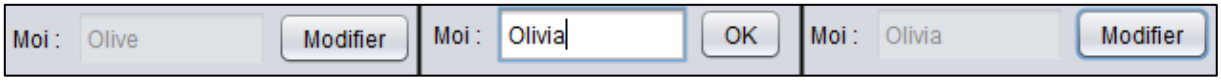
Les différents éléments sont les suivants :

1. Votre pseudonyme visible et la possibilité de le modifier.
2. La liste des utilisateurs actifs avec leurs pseudonymes.
3. Le bouton de déconnexion pour éventuellement changer de compte/utilisateur. Fermer la fenêtre à l'aide de la croix en haut à droite est équivalent à se déconnecter.
4. La zone utilisée lors des sessions de clavardage.

Changer de pseudonyme

Une fois connecté, vous pouvez changer de pseudonyme à n'importe quel moment.
Pour changer votre pseudonyme, cliquez sur le bouton « Modifier » à droite de votre pseudonyme. Cela change le bouton « Modifier » en « OK » et vous permet de saisir un nouveau pseudonyme. Quand vous avez choisi, cliquez sur « OK ». Si aucun autre utilisateur actif n'a le pseudonyme choisi, alors la modification est effectuée et les autres utilisateurs en sont notifiés (votre pseudonyme change dans la liste des utilisateurs actifs). Sinon, votre pseudonyme est inchangé (l'ancien pseudonyme est gardé). Dans tous les cas, le bouton redevient « Modifier » et vous ne pouvez plus effectuer de modification à moins de recommencer la procédure.

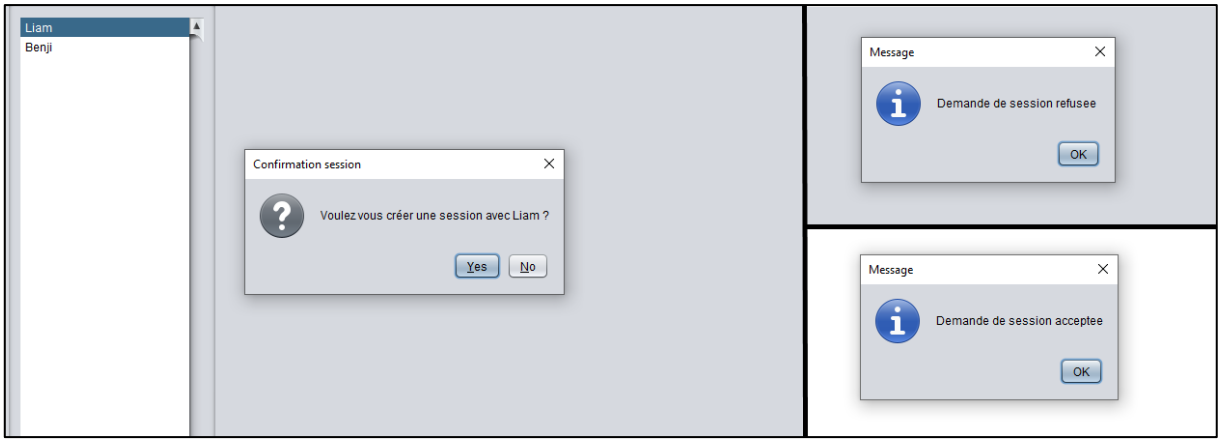
Si vous changez de pseudonyme en pleine session de clavardage, le pseudonyme affiché lors de l'envoi d'un message pour le ou les autres utilisateurs changera aussi. Le pseudonyme de l'onglet de session (voir Session de clavardage) ne changera pas.



Initier une session de clavardage

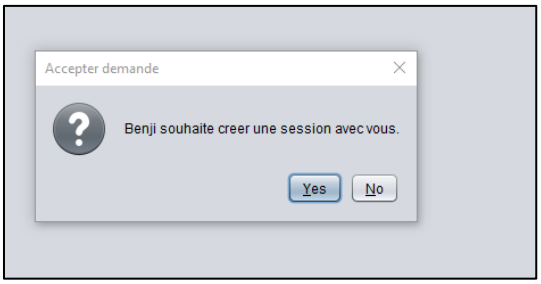
Vous pouvez initier une session de clavardage en cliquant sur le pseudonyme d'un des utilisateurs de la liste des utilisateurs actifs. Une boite de dialogue vous demandera alors si vous souhaitez vraiment démarrer une session avec l'utilisateur choisi. En répondant oui, l'autre utilisateur aura une boite de dialogue similaire qui s'affichera. Si sa réponse est oui, vous en serez informé et la fenêtre de session s'affichera dans la fenêtre principale.

Vous ne pouvez pas créer de session avec un utilisateur avec lequel vous avez déjà une session en cours.



Répondre à une demande de session

Lorsqu'un utilisateur souhaite créer une session avec vous, vous recevez une boite de dialogue afin valider ou non la création d'une session.



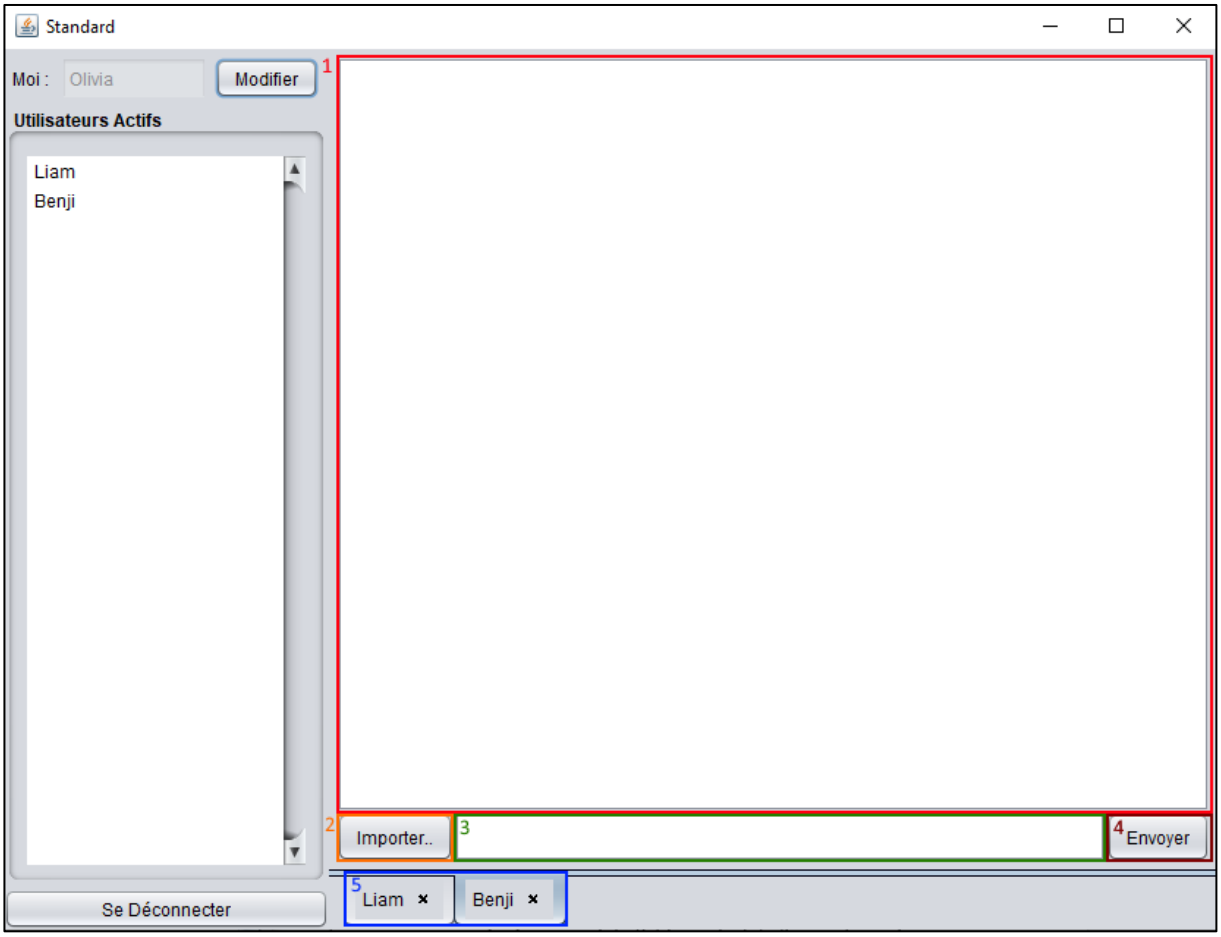
Se déconnecter

Une fois connecté, vous pouvez changer de pseudonyme à n'importe quel moment.
Vous pouvez vous déconnecter soit en cliquant sur le bouton « Se déconnecter », soit en cliquant sur la croix en haut à droite de la fenêtre. Dans tous les cas, la fenêtre principale se ferme et vous revenez à la fenêtre de connexion.

Se déconnecter en pleine session de clavardage revient à y mettre fin (voir Mettre fin à une session de clavardage).

Session de clavardage

Lorsque vous démarrez une session avec un ou plusieurs autres utilisateurs, vous devriez voir apparaître ceci sur la fenêtre principale :



Les différents éléments sont les suivants :

1. Zone de la session de clavardage dans laquelle sont affichés les messages échangés lors des précédentes sessions et les messages de la session actuelle avec éventuellement les miniatures des fichiers échangés lorsqu'il s'agit d'images.
2. Le bouton pour importer des fichiers à envoyer à l'autre utilisateur.
3. La barre de saisie des messages.
4. Le bouton pour envoyer les messages et les fichiers importés.
5. Les onglets de session pour passer d'une session à l'autre et fermer les sessions.

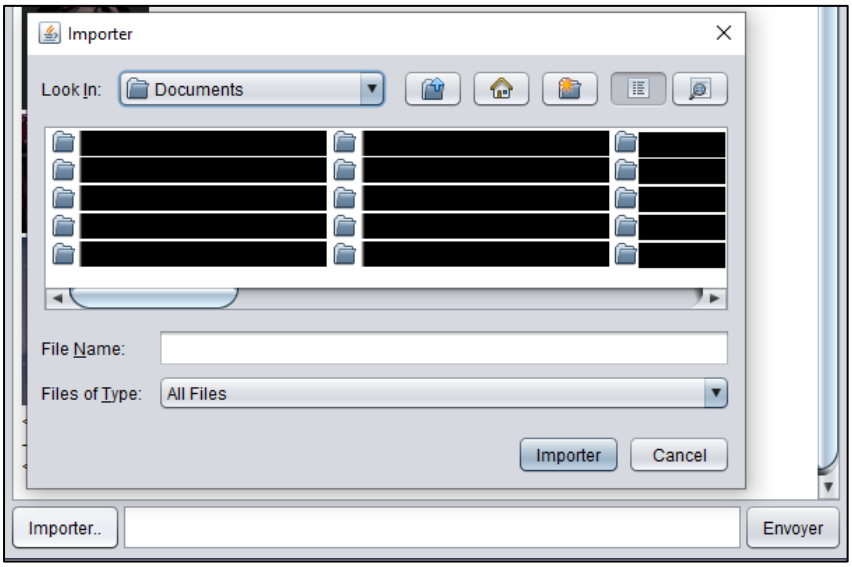
Si vous avez échangé des messages avec le ou les autres utilisateurs lors de précédentes sessions, l'historique de ces messages doit s'afficher dans la zone de session.

Envoyer un message

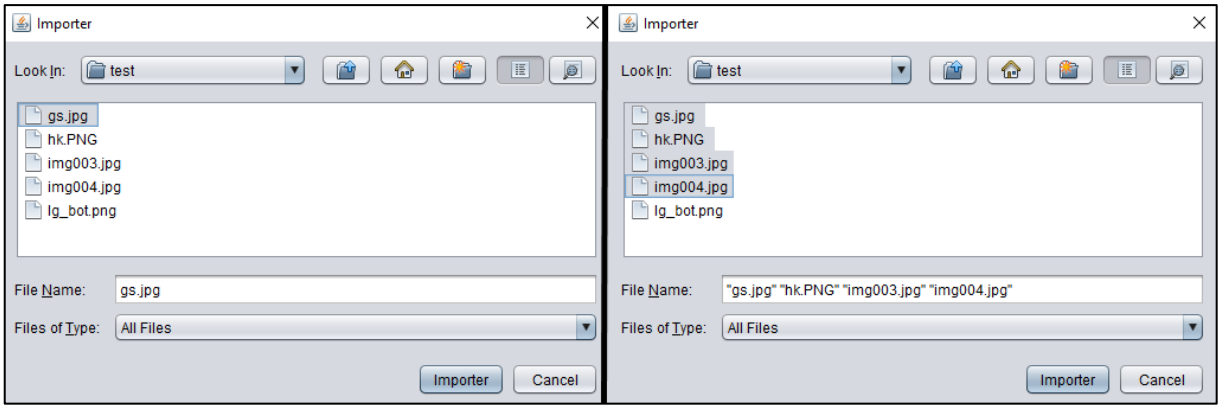
Au sein d'une session de clavardage, vous pouvez envoyer un message simple en l'écrivant dans la barre de saisie. Vous pouvez ensuite envoyer le message soit en cliquant sur le bouton « Envoyer », soit en appuyant sur la touche « Entrée ». Vous avez aussi la possibilité de revenir à la ligne dans un message en appuyant sur shift + Entrée. Shift étant la touche représentée par le symbole ⇧ et située en dessous de la touche de verrouillage des majuscules (à gauche du clavier).

Envoyer un ou plusieurs fichiers

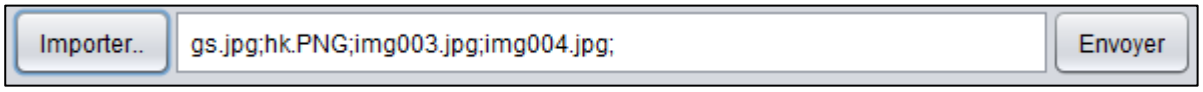
Pour envoyer des fichiers, commencez par cliquer sur le bouton « Importer ». Cela ouvre un explorateur placé par défaut dans votre dossier « Documents ». Vous pouvez ensuite vous déplacer dans d'autres dossiers pour sélectionner un ou plusieurs fichiers à envoyer :



Pour une sélection simple, cliquez simplement sur le fichier que vous souhaitez envoyer.
Pour une sélection multiple, maintenez Ctrl (en bas à gauche de votre clavier) et cliquez sur les fichiers que vous souhaitez envoyer. Cela ne fonctionne que pour des fichiers situés dans le même dossier.
Si vous souhaitez envoyer des fichiers situés dans des dossiers différents, importez-les en plusieurs fois.



Chaque fois que vous importez un fichier, son nom s'affiche dans la barre de saisie du chat suivi du séparateur « ; ». Si vous souhaitez annuler l'envoi d'un fichier, effacez simplement son nom et le séparateur « ; » :



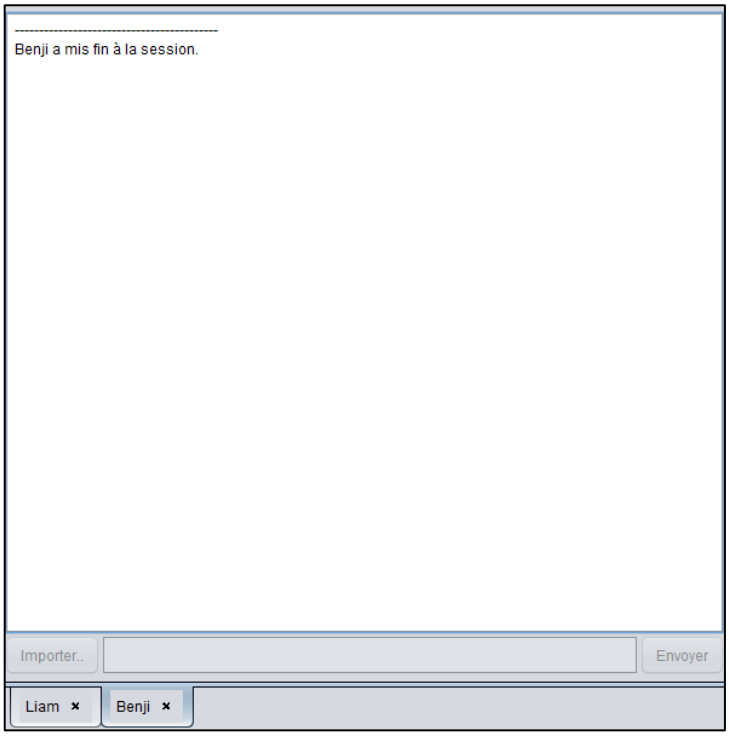
Ne saisissez pas de message à la suite des noms des fichiers que vous voulez envoyer.
Vous pouvez cependant continuer à envoyer des messages **pendant** un envoi de fichiers, c'est-à-dire après avoir appuyé sur « Envoyer » ou sur la touche « Entrée » une fois la sélection de fichiers terminée.

Recevoir un ou plusieurs fichiers

Il n'y a aucune demande de validation lorsque les utilisateurs avec lesquels vous êtes en session vous envoient des fichiers. Lorsqu'un fichier est reçu, un message s'affiche avec le nom du fichier que vous avez reçu. Lorsqu'il s'agit d'une image, une miniature de celle-ci est affichée plutôt que du texte. Tous les fichiers/images reçus sont stockés dans le dossier « downloads/ » situé dans le même dossier que celui de l'application.
Soyez donc prudent avant d'ouvrir un fichier reçu.

Mettre fin à une session de clavardage

Vous pouvez mettre fin à une session de clavardage simplement en cliquant sur la croix de l'onglet correspondant en bas de la fenêtre.
Si vous êtes celui qui ferme la session, l'onglet disparaît.
Si l'utilisateur avec lequel vous communiquez ferme la session, un message s'affiche vous indiquant que l'utilisateur a mis fin à la session et les boutons « Importer », « Envoyer » et la barre de saisie sont désactivés, vous encourageant ainsi à fermer l'onglet.
L'autre utilisateur ne pourra pas initier une nouvelle session avec vous tant que vous n'aurez pas fermé l'onglet de la session précédente (toute demande de création lui sera refusée).



Serveur de présence

Veuillez noter que le serveur de présence ne fonctionne qu'en mode local, avec un système de ports. Au vu des circonstances, il était compliqué d'accéder à des ressources informatiques nous permettant de tester avec un véritable réseau local. De plus, le serveur étant à priori disponible en permanence, nous ne garantissons pas le comportement si le serveur est démarré après l'application.

Une fois le serveur lancé, la page d'accueil s'affichera :

Les quatre modèles de requêtes peuvent être copiées-collées à la suite de l'URL et remplies comme vous le souhaitez. Toutefois, le serveur ne reconnaît actuellement que 3 id : user1, user2 et user3. Toute tentative de connexion avec une autre id provoquera un message d'erreur. Vous recevrez également un message d'erreur si votre requête n'a pas la syntaxe attendue, ou si vous essayez de vous connecter avec un pseudo déjà utilisé. Une fois connecté à l'aide d'une requête POST, la liste des utilisateurs actifs, locaux et externes, s'affichera. Elle n'est pas mise à jour en temps réel, mais peut être rafraîchie manuellement avec une requête GET.

Les utilisateurs lançant la version modifiée de l'application verront, à la connexion, un utilisateur nommé « Serveur de présence », comme montré sur l'image suivante. Cela permet d'attester que le serveur de présence est bien actif. Les utilisateurs externes s'afficheront à la connexion, de la même manière que les utilisateurs locaux. Si vous tentez d'établir une session de clavardage avec le serveur de présence ou un utilisateur externe, celle-ci sera automatiquement refusée.

Choix techniques et implémentation

Environnement de développement

L'application a été entièrement développée sous Eclipse, pour ses nombreuses fonctionnalités pour les projets Java et Java Web. Pour simuler un serveur, nous avons utilisé Tomcat, qui était la solution technique la plus simple.

Interface graphique

Nous avons choisi d'utiliser Java Swing, standard que nous maîtrisions et qui combine grande flexibilité et simplicité d'utilisation.

Communication

La transmission des informations des utilisateurs lors des opérations de connexion, déconnexion et changement de pseudonyme fut une problématique durant l'implémentation de la communication UDP. Après concertation, nous avons décidé d'appliquer la démarche suivante :

- Lorsqu'un utilisateur se connecte et avant de choisir son pseudonyme, l'application envoie un signal de connexion en broadcast sur le réseau local. Le but de ce signalement est de constituer la liste des utilisateurs actifs pour l'utilisateur qui vient de se connecter afin qu'il choisisse un pseudonyme unique. La vérification de l'unicité est donc faite en local.
- Lorsqu'une application reçoit un signalement de connexion, elle répond en envoyant les informations de son utilisateur.
- Lorsqu'un utilisateur choisit son pseudonyme lors de la connexion ou lors d'un changement de pseudonyme (et que celui-ci est valide), l'application envoie les nouvelles informations de l'utilisateur en broadcast sur le réseau local.
- Lorsqu'un utilisateur se déconnecte, l'application envoie un signal de déconnexion en broadcast avec les informations de l'utilisateur sur le réseau local. Si l'utilisateur déconnecté fait partie de la liste des utilisateurs actifs, il en est alors retiré.

En fonction de la bande passante et lors de connexions simultanées de plusieurs utilisateurs, il est possible que différents utilisateurs choisissent le même pseudonyme ou qu'ils ne puissent pas créer de session de clavardage (car absents de leurs listes d'utilisateurs actifs).

Base de données

La technologie retenue est SQLite. La base de données est décentralisée, elle contenue dans un simple fichier local et propre à l'application. Nous avons choisi SQLite pour sa facilité d'installation, d'utilisation, d'accès (pas de dépendance à un serveur), sa portabilité et sa rapidité. Il s'agit de plus d'une technologie adaptée par rapport au volume d'informations échangé et stocké. La maintenance de l'application et le développement de nouvelles fonctionnalités dépendantes de la base de données seront également facilités.

L'architecture de la base de données est la suivante :

- La table « user » contenant toutes les informations nécessaires à l'authentification de l'utilisateur et à la gestion de l'historique des messages.
- La table « conversation » qui permet de représenter le fait qu'un utilisateur envoie des messages à un autre utilisateur.
- La table « message » dans laquelle est stocké le contenu des messages, la conversation à laquelle ils sont rattachés, leur type, leur date d'envoi, éventuellement leur extension.
- La table « type » dans laquelle sont stockés les différents types de messages. Il s'agit d'une table statique qui sert juste à stocker les différents types de messages en tant que constantes, SQLite ne proposant pas le type ENUM.

Sécurité

Les mots de passe utilisateurs ainsi que les messages sont chiffrés dans la base de données pour éviter qu'un éventuel attaquant accède aux mots de passe et/ou messages échangés.
Le chiffrement utilisé est le chiffrement AES/CBC/PKCS5Padding.
Le hachage est réalisé avec la fonction PBKDF2-HMAC-SHA1.
La méthode utilisée pour assurer la sécurité des données est tirée d'un forum d'échange sur le chiffrement des données au sein d'une base de données SQLite : https://security.stackexchange.com/questions/63343/encrypting-data-within-sqlite-database-in-java-how-to-store-key/63869 .
Deux informations sont nécessaires pour créer un utilisateur : Un nom d'utilisateur unique et un mot de passe.

Lors de l'insertion d'un nouvel utilisateur dans la base, plusieurs données sont générées :
- Un sel pour le mot de passe.
- Une clé de chiffrement/déchiffrement des informations relatives à cet utilisateur. Cette clé est générée pour utiliser le chiffrement AES.
- Un sel pour générer une clé qui va servir à chiffrer la clé mentionnée ci-dessus.
- Un vecteur d'initialisation pour utiliser le chiffrement AES-CBC.

Une clé de chiffrement est ensuite obtenue en hachant le mot de passe avec le sel prévu pour sa génération. Cette clé va servir à chiffrer la clé de chiffrement des informations de l'utilisateur.
Le hash du mot de passe (résultat du hachage du mot de passe) est ensuite récupéré et chiffré avec la clé de chiffrement des informations utilisateur et le vecteur d'initialisation.
Cette clé est elle-même chiffrée avec la clé de chiffrement dérivée du mot de passe et le vecteur d'initialisation.
Enfin, on stocke dans la base de données :
- Le nom d'utilisateur.
- Le hash chiffré du mot de passe.
- Le sel du mot de passe en clair.
- Le sel servant à générer la clé de chiffrement en clair.
- La clé de chiffrement des informations utilisateur sous forme chiffrée.
- Le vecteur d'initialisation en clair.

La procédure pour authentifier un utilisateur est ensuite la suivante :
- Récupérer les données de l'utilisateur.
- Générer une clé de chiffrement à partir du mot de passe et du sel correspondant.
- Déchiffrer la clé de chiffrement des informations utilisateur avec la clé obtenue précédemment et le vecteur d'initialisation.
- Déchiffrer le hash chiffré du mot de passe avec la clé précédemment déchiffrée et le vecteur d'initialisation.
- Hacher le mot de passe rentré par l'utilisateur avec le sel du mot de passe.
- Comparer le hash obtenu avec le hash déchiffré.

Si les deux hash sont égaux, le mot de passe est correct. Sinon le mot de passe est incorrect.

Les messages sont eux aussi chiffrés. La première fois que deux utilisateurs communiquent, deux entrées dans la table conversation sont crées à la fin de leur session. Un vecteur d'initialisation est créé pour chacune des conversations.
Le contenu des messages est ensuite chiffré avec la clé de chiffrement des informations utilisateur et ce vecteur d'initialisation.

Sauvegarde des messages

Le choix retenu est de sauvegarder tous les messages à la fin de la session en utilisant une transaction (autocommit désactivé) à des fins d'optimisation. Se déconnecter ou fermer la fenêtre principale avec une ou plusieurs sessions en cours est équivalent à terminer la(les) session(s). A moins d'un bug ou crash de l'application, le fait que les messages échangés ne soient pas sauvegardés est peu probable.

Serveur de présence

Le serveur comporte une liste d'Utilisateur, remoteUsers, qui remplit le même rôle que la liste des utilisateurs dans l'application classique, mais pour les utilisateurs externes. En effet, les utilisateurs externes n'interagissent avec le serveur que par requêtes HTTP, il serait donc contre-intuitif de les mettre dans la même liste.

Le serveur de présence dispose également du même système de communication UDP que n'importe quel utilisateur, y compris la liste des utilisateurs locaux. Cela lui permet d'interagir avec le réseau local comme un utilisateur lambda. Le serveur traite donc tous les messages concernant des changements dans la liste d'utilisateurs en provenance du réseau local, et les répercute sur la liste des utilisateurs locaux.

Le serveur traite également les requêtes HTTP qui lui sont adressées. Toutes ces requêtes sont en fait des requêtes GET déguisées à l'aide du paramètre de requête type, et aiguillées vers les fonctions doPost(), doPut et doDelete() si nécessaire. L'intention initiale était de mettre en place une véritable interface graphique à l'aide du langage JSP, mais le temps et les compétences nécessaires nous ont fait défaut. Le serveur transmet aux utilisateurs externes les diverses informations nécessaires (page d'accueil, messages d'erreur, liste des utilisateurs connectés) via la réponse HTTP, sous forme de pages html.

Une fonction nommée snotify() (abrégé de server-notify, le mot-clé notify étant réservé) permet d'informer les utilisateurs locaux des modifications apportées par les utilisateurs externes. Le serveur assume ici un rôle de proxy, en traduisant toutes les modifications apportées via HTTP en signal qu'il diffuse ensuite sur le réseau local (ou, dans notre cas, sur les ports enregistrés), en y appliquant son propre numéro de port. Si le serveur venait à évoluer vers un serveur supportant le chat via TCP, on peut supposer qu'il jouera là aussi un rôle de proxy.

La fonction snotify() est appelée automatiquement à chaque modification apportée via HTTP. Nous avons considéré appeler la fonction snotify uniquement manuellement par une requête GET, mais cela posait des problèmes au niveau de la synchronisation du choix des pseudo. En effet, cela rendait possible qu'un utilisateur externe choisisse un pseudo et n'en informe pas les utilisateurs locaux, et qu'un utilisateur local choisisse le même pseudo. Lorsque l'utilisateur externe appellerait alors snotify(), deux utilisateurs auraient alors le même pseudo, ce que le cahier des charges interdit.

Notes et améliorations possibles

- Compléter les sources pour faire fonctionner l'application sur un réseau local.
- Implémenter un outil pour remplacer une table d'une base de données par une autre.
- Ajouter des boutons radio à l'écran de connexion pour différencier utilisateurs internes/externes.
- Améliorer l'envoi et la réception de fichiers en termes de sécurité. En l'état, pas de validation requise pour recevoir des fichiers.
- Gérer la réception de fichiers de deux utilisateurs différents ayant les mêmes noms et extensions. En l'état, le fichier est stocké dans le dossier downloads et est écrasé lors de la réception d'un fichier du même nom et extension.
- Lier le serveur de présence à la base de données et mettre en place un véritable mécanisme d'authentification. Nous avons essayé de mettre ceci en place, mais à la suite d'un problème d'importation des librairies SQLite que nous n'avons pas su résoudre, nous avons préféré rendre une version précédente et fonctionnelle du code.
- Garder en mémoire les informations de l'utilisateur externe à l'aide des cookies, pour prévenir l'usurpation d'identité et rendre les interactions plus ergonomiques.
- Mettre en place une véritable interface graphique (par exemple à l'aide du langage JSP), ce qui, en plus de rendre le serveur plus simple à utiliser, évitera que des utilisateurs non authentifiés puissent avoir accès à la liste des utilisateurs actifs.

INSA Toulouse
135, avenue de Rangueil
31077 Toulouse Cedex 4 - France
www.insa-toulouse.fr

Université Fédérale Toulouse Midi-Pyrénées
Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
MINISTÈRE
DE L'ÉDUCATION NATIONALE,
DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE