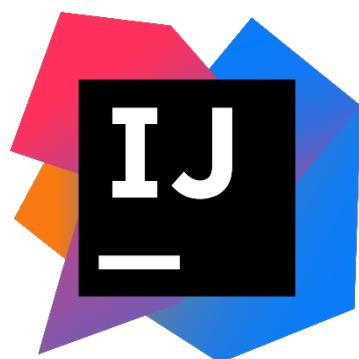


Anisse FOUKA ; Yanis ZERRAR ;

Mehdi BERRADA

Compte-rendu SAÉ :

SAÉ 32 : Découvrir des applications
communicantes



Professeurs : Spies François

Table des matières

I.	Présentation de la SAE	3
A.	Introduction.....	3
B.	Organisation du groupe	4
C.	Matériel et environnement de développement pour le projet de messagerie UDP.....	6
1.	Serveur et Environnement de Développement :	6
II.	Fonctionnement de l'application	7
A.	Jframe.....	7
1.	Objectifs et Fonctionnalités du Client JFrame :.....	7
2.	Interface Graphique Basique :.....	7
3.	Connexion et Communication :.....	7
B.	Android studio	8
1.	Fonctionnement d'Android Studio :.....	9
2.	Conception de l'Application :	9
3.	Écran de Connexion et Inscription :	9
4.	Ajout d'Amis et Conversation :.....	10
III.	Serveur UDP.....	11
A.	Stockage de données	11
IV.	Conclusion :.....	12

I. Présentation de la SAE

A. Introduction

L'initiative éducative connue sous l'appellation SAé se réfère à une Situation d'Apprentissage et d'Évaluation.

Spécifiquement, la « SAé 302 : Développer des applications communicantes » s'est déroulée du lundi 11 au vendredi 15 décembre 2023 au sein de l'Institut Universitaire de Technologie spécialisé en Réseaux et Télécommunications situé à Montbéliard. Ce projet avait pour but la conception d'une application de type client/serveur en s'appuyant sur des fonctions de communication pour élaborer un protocole applicatif par-dessus UDP.

Les buts pédagogiques de cette SAE incluaient l'application de techniques de gestion de projet pour la réalisation efficace de cette tâche. Cela impliquait l'usage du langage Java, tel qu'approfondi dans le module R308 : Consolidation Programmation, afin de concevoir et d'implémenter une communication entre client et serveur via les protocoles TCP ou UDP. Nous utiliserons UDP

Ce compte-rendu de projet va mettre en lumière les stratégies de gestion de projet que nous avons adoptées ainsi que la mise en œuvre du modèle client/serveur que nous avons utilisé.



B. Organisation du groupe

Répartition et Organisation du Travail au Sein du Groupe pour la SAE 302

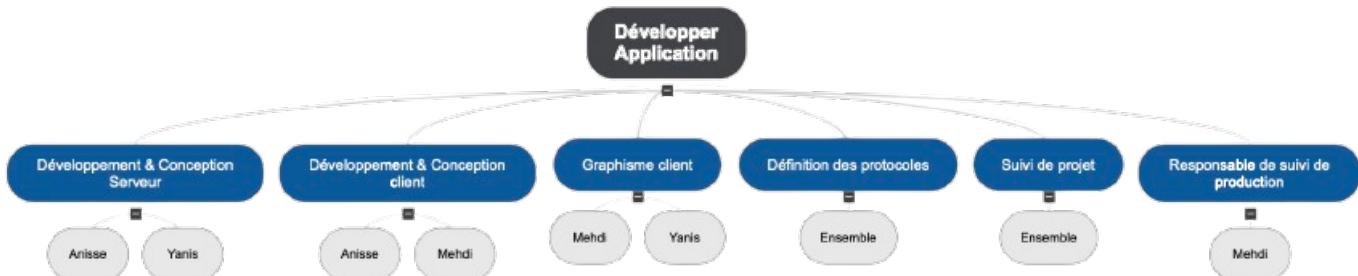
Pour le projet SAE 302, notre groupe était composé de Yanis Zerrar, Mehdi Berrada, et moi-même, Anisse Fouka. Nous nous sommes associés principalement en raison de nos compétences complémentaires et de notre enthousiasme commun pour le développement d'applications communicantes. La cohésion s'est rapidement établie entre nous, favorisée par une bonne entente et une volonté partagée de réussir.

Nous avons veillé à ce que la gestion de notre groupe soit équitable et efficace. Chacun de nous avait des responsabilités clairement définies, choisies en fonction de nos affinités et de notre expertise respective. Yanis a pris en main la structuration du code, apportant sa précision et son attention aux détails. Anisse, avec sa capacité à résoudre les problèmes et son aisance avec les concepts de réseau, s'est concentré sur l'implémentation des protocoles UDP. De mon côté, Mehdi, j'ai endossé le rôle de coordonnateur, m'assurant que le projet avançait selon le planning et que la collaboration restait fluide couplé bien évidemment à la création de l'application Android Studio

L'équité dans la répartition des tâches nous a permis de travailler dans un esprit de solidarité et d'entraide, où chacun pouvait compter sur le soutien des autres pour accomplir sa part. Cela a contribué à une expérience d'apprentissage enrichissante pour nous tous.

Nous avons tenus à créer une carte mental représentant les différents tâches de manière clair sur les tâches de chacun d'entre nous





C. Matériel et environnement de développement pour le projet de messagerie UDP

Pour mener à bien notre projet de messagerie en utilisant le protocole UDP, nous avons utilisé le PC de Yanis comme serveur central et comme station de développement. Voici les spécifications et l'usage de ce matériel dans le cadre de notre projet :

1. Serveur et Environnement de Développement :

Le PC de Yanis, fonctionnant sous Windows 11, doté de 16 Go de RAM, ce qui a largement contribué à la fluidité et à la performance lors du développement et des tests.

Pour le développement initial du serveur et du client de bureau, nous avons utilisé l'IDE IntelliJ IDEA. Cela nous a permis d'écrire, de tester et de déboguer le code de notre application en Java de manière efficace.

Développement de l'Application Mobile :

Après avoir établi les bases de notre code sur IntelliJ IDEA, nous avons poursuivi avec le développement d'une application mobile à l'aide d'Android Studio.

Cette transition vers le développement mobile a été réalisée pour adapter notre projet à une application cliente qui peut s'exécuter sur des appareils Android.

L'application mobile a été testée sur un émulateur intégré dans Android Studio ainsi que sur des appareils mobiles réels pour s'assurer de sa compatibilité et de sa communication réussie avec le serveur via le protocole UDP.

L'utilisation du PC de Yanis comme serveur et comme environnement de développement a été cruciale, car elle nous a permis de disposer d'une plateforme stable et performante pour le cycle complet de notre projet, de la conception à la réalisation de l'application cliente mobile.

II. Fonctionnement de l'application,

A. Jframe

Le client JFrame, conçu spécifiquement pour les premiers tests du serveur de chat, a été un élément crucial dans la validation de la communication entre les utilisateurs et le serveur. Son développement initial visait à évaluer la capacité du serveur à faciliter les échanges de messages entre plusieurs clients. Ce client JFrame offrait une interface utilisateur simplifiée pour des interactions de base, telles que la saisie et l'envoi de messages.

1. Objectifs et Fonctionnalités du Client JFrame :

L'objectif principal était d'assurer la communication fluide entre les clients et le serveur. Pour ce faire, le client JFrame a été doté des fonctionnalités suivantes :

2. Interface Graphique Basique :

L'interface minimale comportait des zones de texte pour la saisie des messages et l'affichage des réponses.

Un bouton d'envoi était inclus pour transmettre les messages saisis vers le serveur.

Validation de la Communication :

Le client JFrame permettait d'établir une connexion au serveur en spécifiant le nom d'hôte pour initier les échanges de messages.

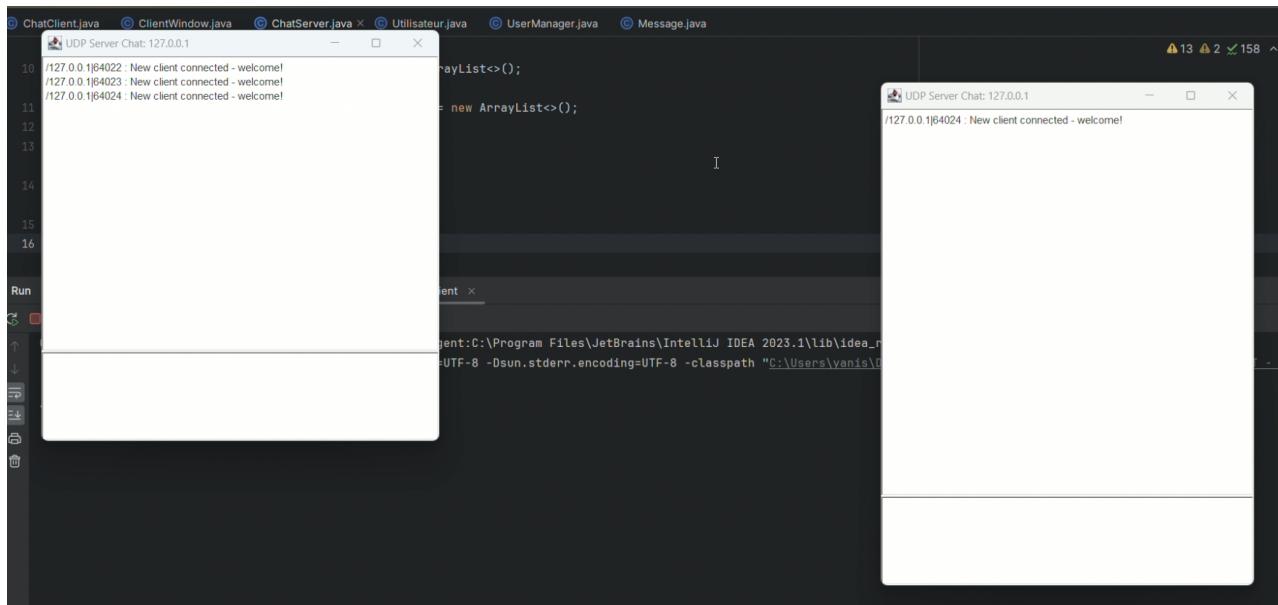
Il envoyait les messages saisis par l'utilisateur au serveur et affichait les réponses reçues dans l'interface.

Dans le processus de test, ce client JFrame a été crucial pour valider plusieurs aspects fondamentaux du serveur :

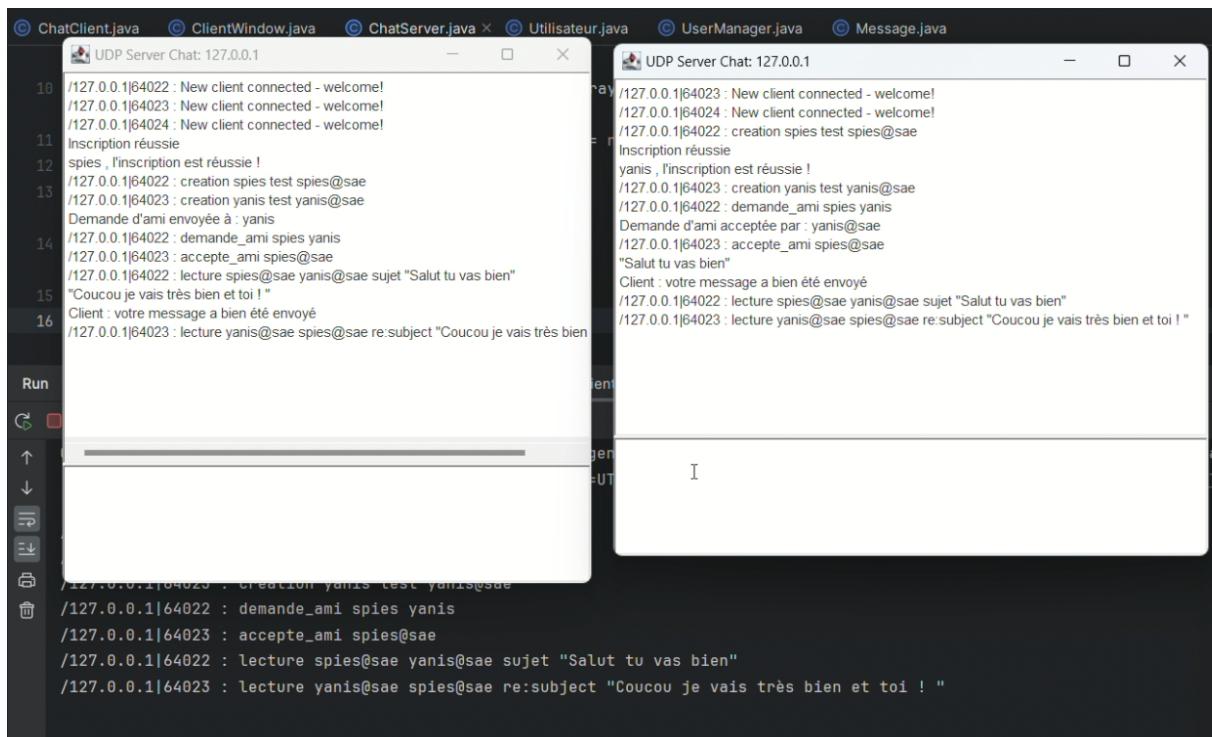
3. Connexion et Communication :

Vérification de la connexion réussie au serveur et confirmation de la capacité à envoyer et recevoir des messages.

Captures d'Écran pour Illustrer la Communication :



Des captures d'écran entre deux clients utilisant ce client JFrame ont été réalisées. Ces captures ont été obtenues pour démontrer visuellement la communication entre les utilisateurs via le serveur.



Le développement initial du client JFrame a été primordial pour les tests et la validation du serveur de chat. Les captures d'écran, illustrant les interactions entre deux clients, ont joué un rôle crucial pour démontrer le bon fonctionnement du système de communication client-serveur. Ces preuves visuelles renforcent l'efficacité et la fiabilité du système, confirmant ainsi la réussite des échanges de messages entre les utilisateurs connectés au serveur.

B. Android studio

Android Studio a été notre choix pour le développement initial de l'application, visant à offrir une expérience de messagerie UDP convaincante sur mobile, bien qu'elle ne soit pas encore entièrement fonctionnelle.

1. Fonctionnement d'Android Studio :

Android Studio est l'environnement de développement officiel pour les applications Android. Il intègre des outils de développement tels que des émulateurs pour tester les applications, un débogueur pour résoudre les problèmes, et un éditeur de code pour écrire et modifier le code source. Avec Android Studio, nous avons pu visualiser les changements en temps réel grâce à l'aperçu de mise en page et tester l'application sur une gamme de tailles d'écran et de versions d'Android.

2. Conception de l'Application :

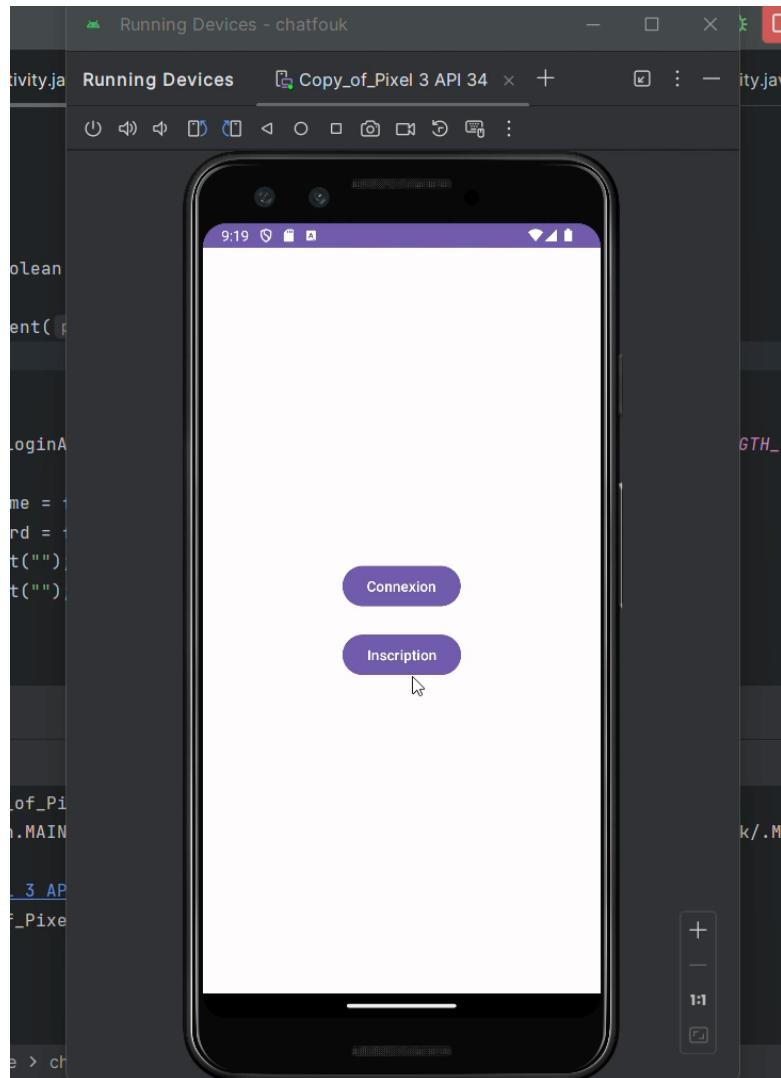
Lors de la conception de l'application dans Android Studio, nous avons d'abord réfléchi à l'expérience utilisateur. L'idée était de créer une interface simple et intuitive. Dès le lancement de l'application, l'utilisateur est accueilli par un écran de démarrage présentant deux options : connexion et inscription. Pour réaliser cela, nous avons utilisé les Activités d'Android pour gérer les écrans et les Intents pour naviguer entre eux.

3. Écran de Connexion et Inscription :

Sur l'écran de démarrage, l'utilisateur peut choisir de se connecter à un compte existant ou de créer un nouveau compte en cliquant sur le bouton d'inscription.

L'inscription requiert l'entrée d'informations de base et, une fois le compte créé, les données sont enregistrées dans une base de données sécurisée.

La connexion mène à l'écran principal de l'application où les fonctionnalités de messagerie deviennent accessibles.

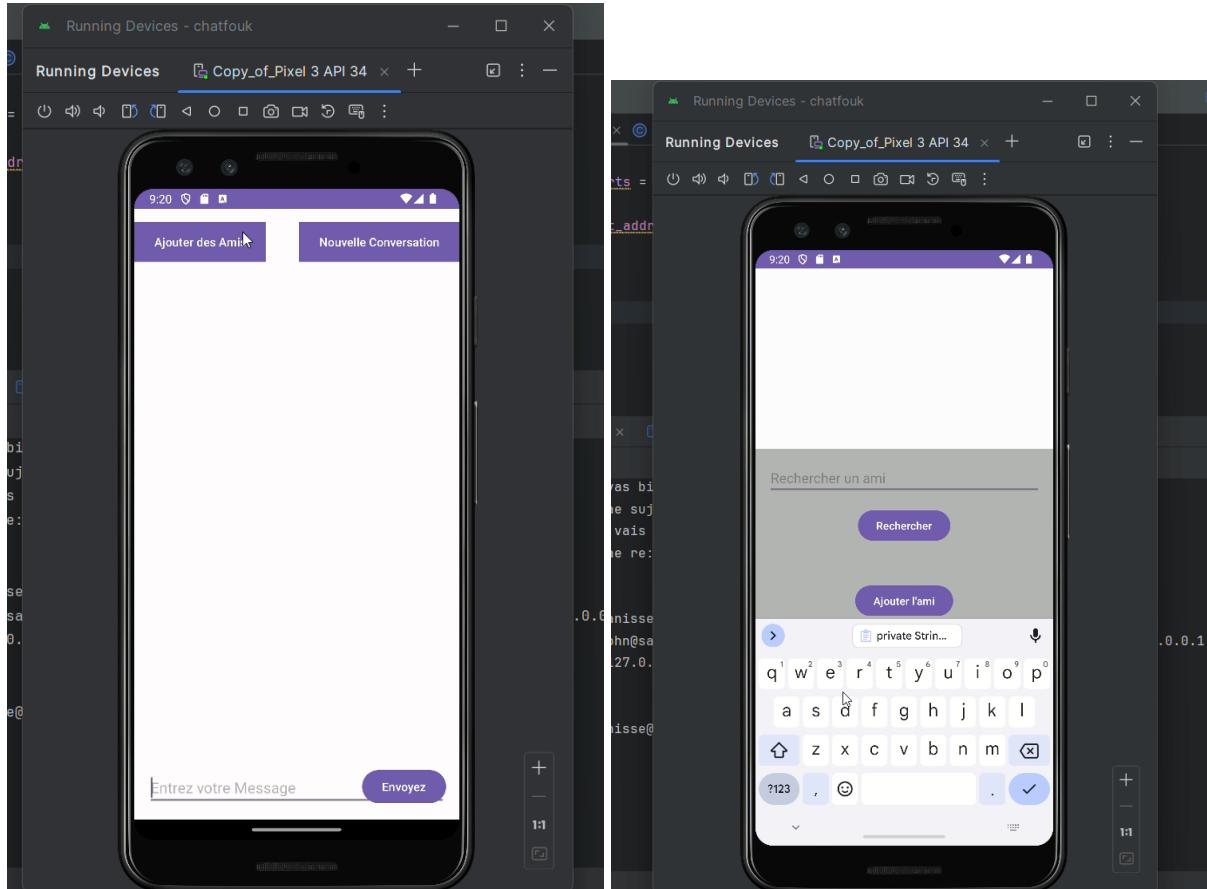


4. Ajout d'Amis et Conversation :

Une fois connecté, l'utilisateur peut ajouter des amis en utilisant leur identifiant unique. Cette fonctionnalité est gérée par un composant de recherche qui interroge le serveur via UDP pour trouver des utilisateurs.

Après avoir ajouté des amis, l'utilisateur peut débuter une conversation. L'interface de messagerie permet d'envoyer et de recevoir des messages en temps réel, avec une gestion des messages via UDP pour assurer une communication rapide et légère.

Le développement avec Android Studio a permis une implémentation efficace de ces fonctionnalités, offrant aux utilisateurs une expérience de messagerie fluide et conviviale sur leurs appareils mobiles.



III. Serveur UDP

A. Stockage de données

Types de Données Stockées :

Liste des Messages : Elle conserve l'historique des messages échangés entre les utilisateurs. Chaque message est stocké avec des détails tels que l'expéditeur, le destinataire, le timestamp et le contenu du message.

Liste d'Amis : Cette liste maintient les relations d'amitié entre les utilisateurs. Elle permet à l'utilisateur de voir qui est en ligne et d'initier des conversations.

Liste des Utilisateurs (Liste User) : Contient les informations de profil de chaque utilisateur, comme les identifiants, les mots de passe (idéalement sous une forme sécurisée), et d'autres métadonnées associées à chaque compte.

Liste des Utilisateurs Bloqués (Liste Bloqués) : Répertorie les utilisateurs qui ont été bloqués par d'autres, empêchant ainsi toute communication indésirable.

Une classe UserManager est responsable de la gestion de toutes les interactions avec ces listes. Elle offre des méthodes pour ajouter, retirer, ou chercher des utilisateurs dans les listes, ainsi que pour gérer les messages et les blocages.

Stockage en Mémoire :

Les données sont stockées sous forme de listes en mémoire vive sur le serveur. Cela permet un accès rapide aux données et une réactivité immédiate de l'application.

Pour assurer la persistance des données en cas de redémarrage du serveur, nous pourrions envisager l'utilisation d'une base de données ou d'un système de fichiers pour enregistrer les informations de manière plus durable.

Cette structure de données est conçue pour optimiser les opérations de l'application de messagerie, en assurant un traitement rapide des données et en offrant une expérience utilisateur fluide et efficace.

IV. Conclusion :

L'application de chat rencontre actuellement des difficultés dans l'affichage des messages reçus du serveur sur l'interface Android Studio, malgré les vérifications et les logs ajoutés pour identifier l'origine du problème. Nous avons le même problème pour afficher la liste de l'amis de l'utilisateur connecté.

La vulnérabilité inhérente à UDP, exposant les risques de DDoS, de spoofing et d'exploitation des services vulnérables, souligne la nécessité cruciale d'implémenter des mesures de sécurité rigoureuses pour contrer les attaques potentielles et garantir l'intégrité des données échangées via ce protocole.

Voici un lien dans lequel nous avons implémenter 2 vidéos de démonstration :

<https://filesender.renater.fr/?s=download&token=3ed06a16-413e-472d-bdea-06881ddc95df>