PAQUELET Etienne POUT Xavier RABERGEAU Nicolas

SAE 302 : Développer des applications communicantes.
Compte rendu

Monsieur SPIES – Du 11/10/2023 au 15/12/2023

R&T
DUT Réseaux & Télécommunications
IUT Belfort-Montbéliard

TABLE DES MATIÈRES..

1. INTRODUCTION A LA SAE.

Une SAE est une Situation d'Apprentissage et d'Évaluation.

La SAE 302 « Développer des applications communicantes » à débuté le lundi 11 décembre 2023 et s'est terminé le 15 décembre 2023.

l'Objectif de ce projet était le suivant :

- créer une application en java utilisant des fonctions de communication pour créer un protocole applicatif au dessus de la pile TCP/IP.

Pour réaliser cet objectif, nous avons créer une application de type de réseau social possédant différentes fonctionnalités telles que :
- Inscription
- Connexion
- Demande d'amis
-Envoi de message : en mode « temps réel si l'utilisateur est connecté » ou en mode « boite au lettre ».
- Chaque compte possède au maximum 10 amis.
- Il ne peut pas y avoir plus de 100 utilisateurs.
- les messages sont stockés en RAM.

Figure 1 : notre travail de groupe.

Afin de réaliser ce projet, nous avons mis en place une méthode de gestion de projet dite «AGILE». Nous avons effectuer une distribution des rôles au début de la SAE afin d'avancer chacun de son coté pour que le projet soit réaliser en temps et en heure.

Etienne
Back-end
Server
Conception de l'architecture
Conception serveur UDP
test de l'évènement et correctif
création requête inscription
création requete connexion sur serveur
création requête demande ami sur serveur
creation envoi message

Nicolas
Génerale
Client/Server
définition protocole de communication
évènement 1 : envoi d'un message au serveur
Conception partie Gestion des évènements du module MVC
création requête demande ami sur client
creation envoi message

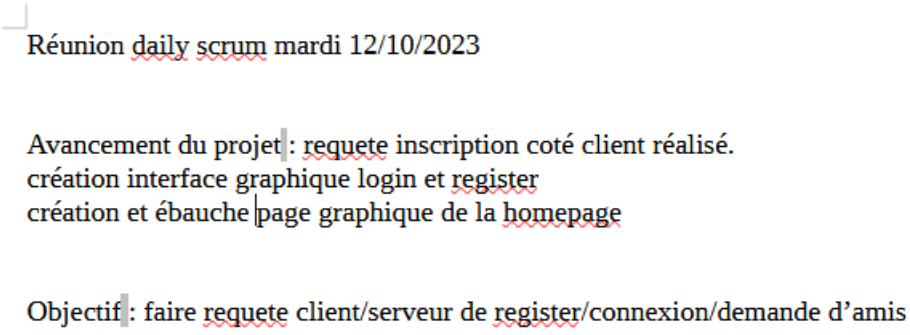
Xavier
Front-end
Client/ GUI
Conception partie Vue du modèle MVC
évènement 1 : envoi d'un message au serveur
Conception GUI client
création requête inscription sur client (récupération donnée et envoi donnée au serveur)
création requête connexion sur client (récupération donnée et envoi donnée au serveur)
creation de envoi message

Voici notre répartition des rôles pour ce projet. Comme on peut le voir, Xavier s'est principalement occupé de l'interface H/M (Homme Machine), Nicolas aidait lorsqu'un membre rencontrait un problème, et Etienne faisait le code niveau serveur. Cependant, c'était les rôles au début du projet, et nous avons successivement échangé les rôles afin d'effectuer un roulement.

Nous avons également réalisé un daily scrum, une réunion journalière, où on faisait le point entre ce qui a été fait la veille, et ce qui doit être fait dans la journée. Nous avons appliqué cette méthode tous les jour et elle nous a

permis de se retrouver dans le projet, où nous en sommes, et qu'est-ce qui nous reste à faire.

Voici le daily scrum de la journée de mardi :


Réunion daily scrum mardi 12/10/2023

Avancement du projet : requete inscription coté client réalisé.
création interface graphique login et register
création et ébauche page graphique de la homepage

Objectif : faire requete client/serveur de register/connexion/demande d'amis

Figure 2 : daily scrum de mardi

3. PRÉSENTATION DE L'ARCHITECTURE TYPE ET DE NOTRE ARCHITECTURE.

Il nous était proposé d'utiliser deux protocoles de communications en début de cette SAE pour faire communiquer le serveur et le client. Nous pouvions choisir entre TCP (Transport Control Protocol) et UDP (User Datagram Protocol). Nous avons décidé d'utiliser le protocole UDP pour différentes raisons :

- simplicité de la syntaxe et rapidité de l'envoi et de la réception des sockets. Nous avons également choisi UDP plutôt que TCP en raison de la manière dont les protocoles communiquent.

- UDP offre une transmission de données sans connexion, ce qui facilite la gestion du programme et évite les erreurs liées à l'ouverture, au maintien et à la fermeture des connexions.

Afin de réaliser ce projet, nous avons tout d'abord réaliser une phase de conception où nous définissions les principes mêmes de l'application que nous devions réaliser. Notre architecture ressemble à l'architecture suivante, où le serveur et le client se trouve sur un même machine, mais le client se connecte au port défini par le serveur UDP.

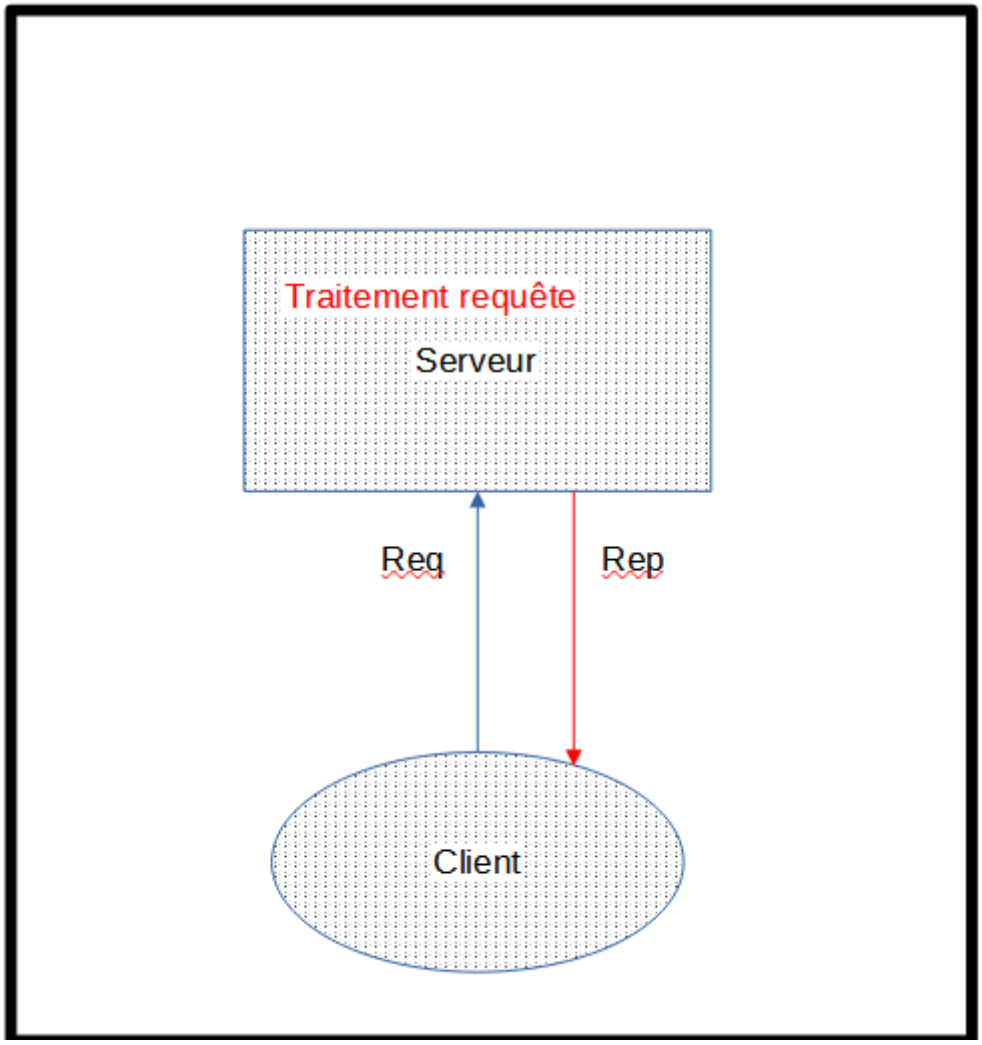


Figure 3 : Notre architecture.

Dans cette architecture, le client envoi la requête sur le port du serveur (vu que le client et le serveur sont sur la même adresse, pas besoin d'utiliser des adresses IP). Le serveur traite la requête, et renvoi une réponse au client en fonction de la sortie du traitement de la requête, si celle ci a fonctionné ou pas.

Cependant, l'architecture type de cette application ressemblerait plus à la figure suivante.

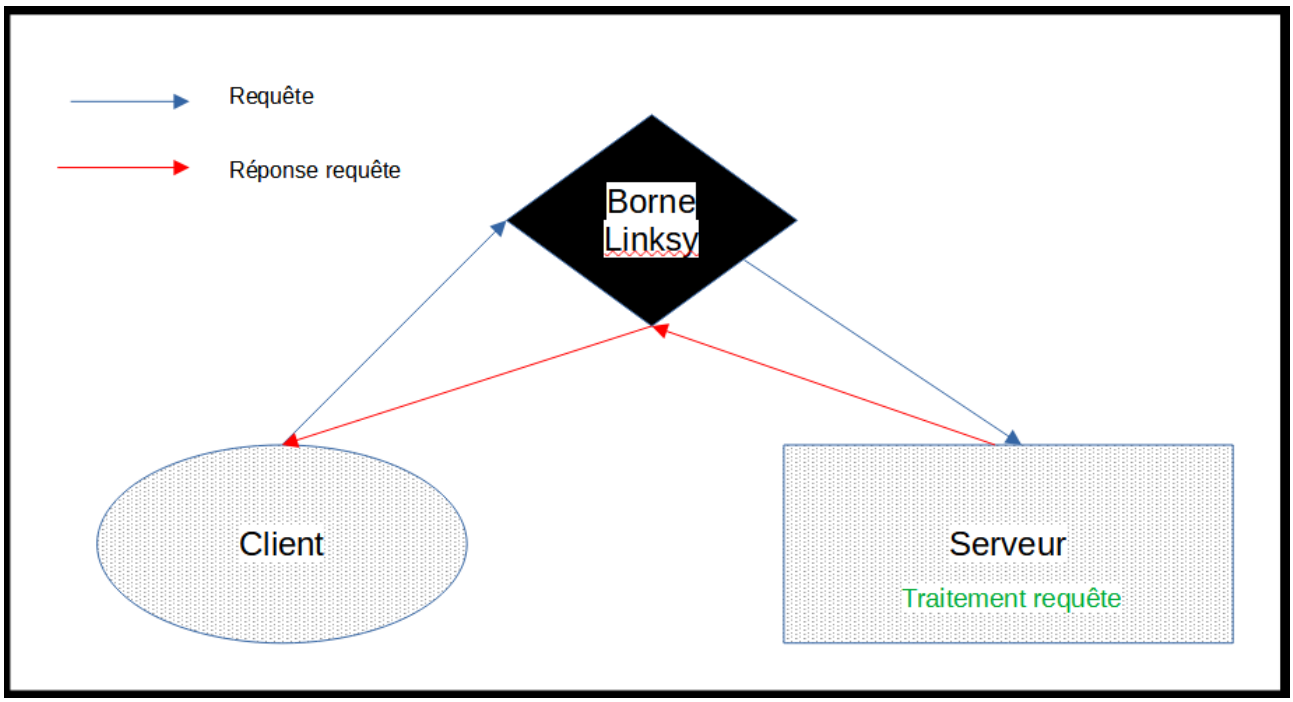


Figure 4 : Architecture type de l'application

L'architecture type de l'application ressemblerait plus à ce modèle. En effet, une application réseau fonctionnant au dessus d'une pile TCP/IP, son but est donc de communiquer entre client et serveur dans un réseau. L'architecture ci-dessus serait donc une architecture plausible pour ce type d'application.

Pour s'inscrire, se logguer, demander un ami, envoyer un message à un ami, le client envoi une requête grâce à des sockets au serveur lui demandant d'effectuer une action. Cependant, comment différencier les actions en fonctions des requêtes ? Pour réaliser cette différentiation des requêtes et des actions associés, nous avons mis en place un protocole de communication de la forme suivante :

typedel'action,donnée,donnée

Ainsi, pour une demande d'inscription, la requête du client au serveur enverra « creation,username,password » et le serveur, dès qu'il reçoit cette requête, si celle-ci contient « creation », alors, il l'enregistrera dans sa base de donnée et enverra une réponse qui contiendra « inscriptionreussie, blablabla » et le client affichera « register done » si il reçoit cette requête.

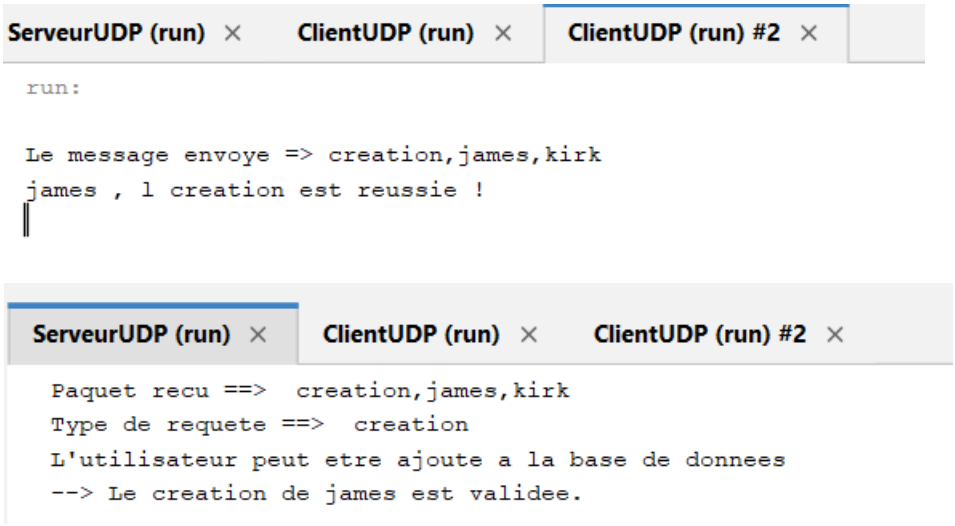


Figure 5 : message des logs lorsque le paquet « creation,james,kirk » est envoyé du client au serveur.

A. Stockage de donnée

Pour réaliser ce projet, nous avons tout d'abord pensé à utiliser une base de donnée MySql afin de stocker les données des utilisateurs lors des requêtes d'inscriptions ou de connexion. Cependant, nous n'avons pas mis en place cette solution, pour la raison que cette solution ne correspondait pas au cahier des charges. Nous avons donc décidé de stocker les données envoyées grâce au protocole de communication dans des tableaux afin de les stocker en RAM. Nous avons utilisé une classe Utilisateur qui stockait toute les données relatives aux utilisateurs (e.g : demande d'amis, amis...) et une classe message qui stockait les messages des Utilisateurs.

B. Modèle graphique

Nous avons réalisé 4 GUI (Graphical User Interface) : 1 login, 1 register, 1 pour ce qui est relatif aux demandes d'amis, liste d'amis, et une correspondant à l'envoi et réception de message. Voici les photos :

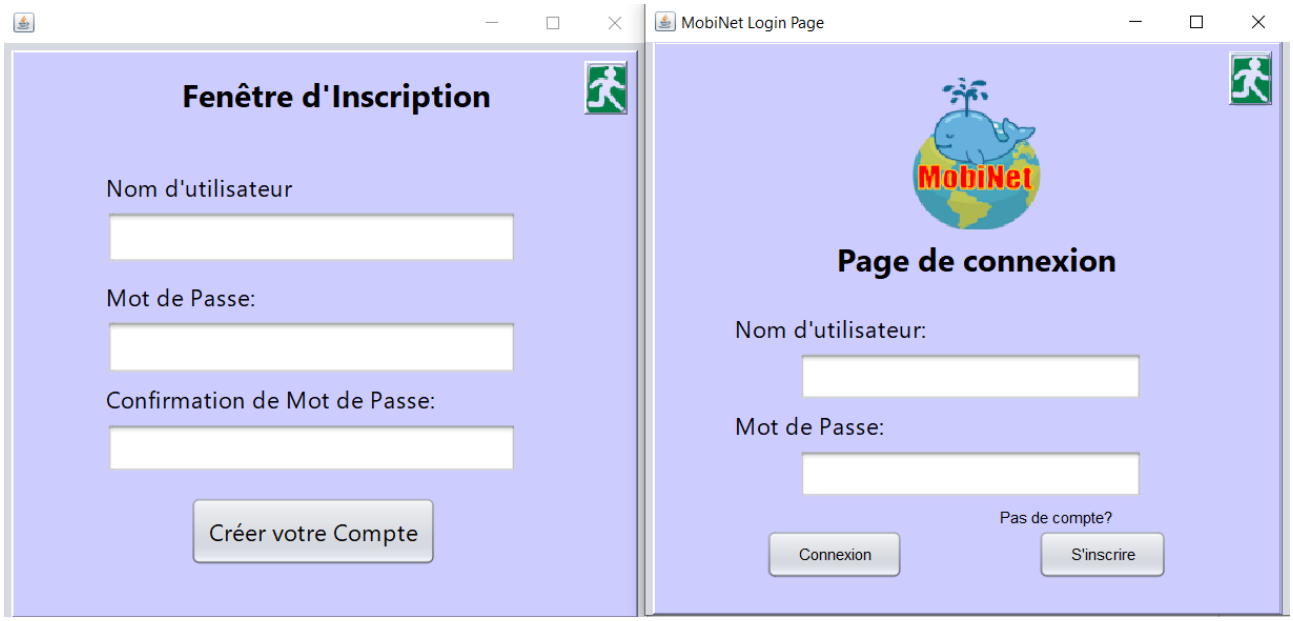


Figure 6 : Page de login et de register.

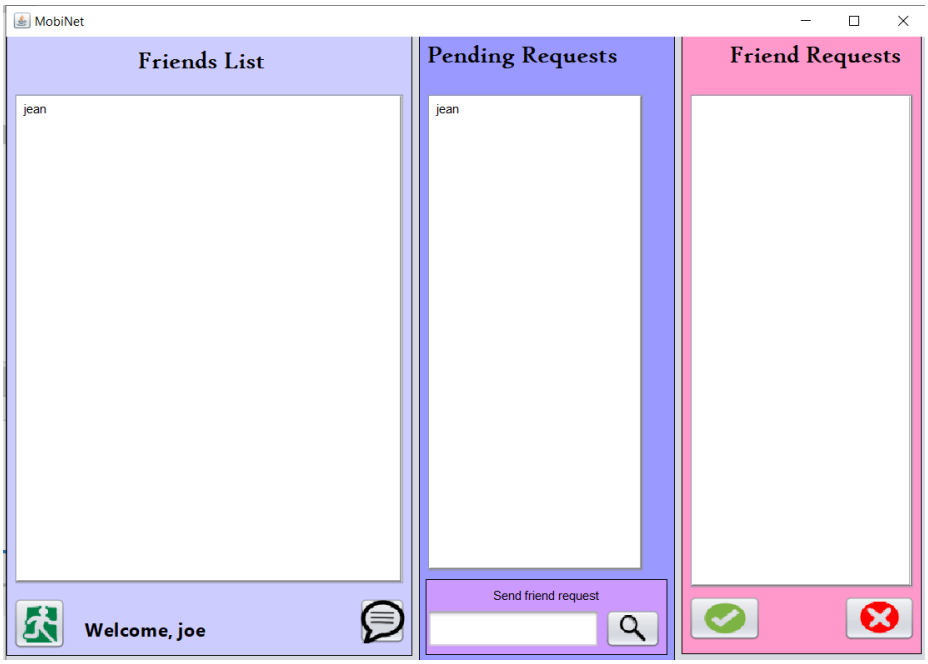


Figure 7 : Notre page Home

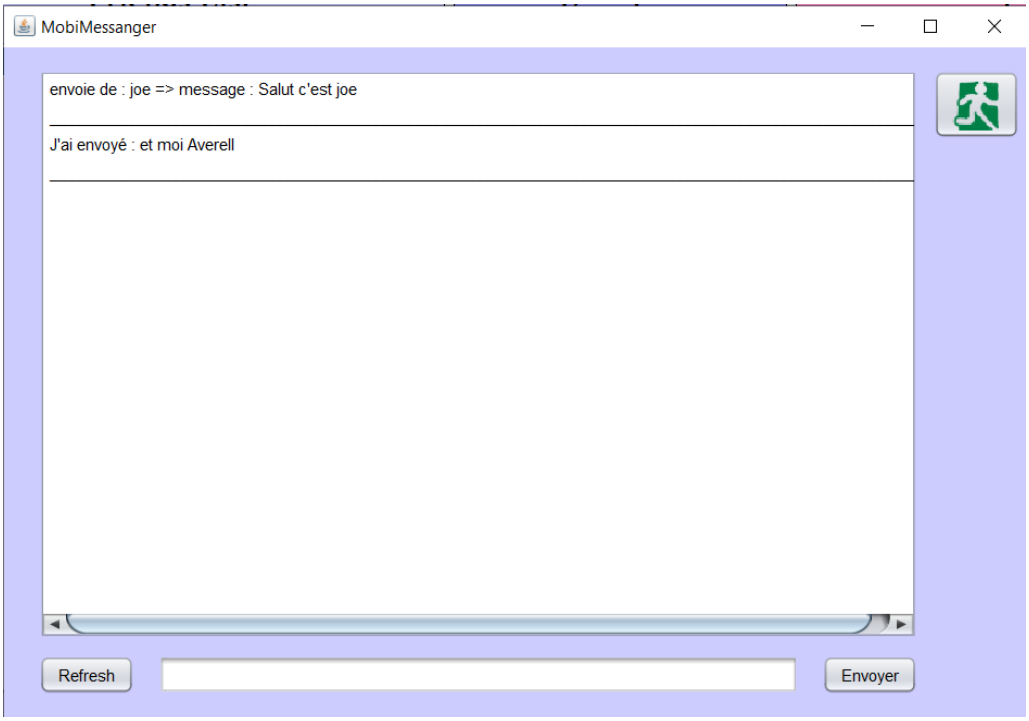


Figure 8 : notre page de messagerie

6. CONCLUSION

Lors de cette SAE, nous avons mis à contribution plusieurs de nos ressources actuelles et de nos compétences : Communication Réseaux, Programmation java, Gestion de Projet et Anglais (pour lire les documentations lorsque nous avions rencontré des problèmes).

Durant cette SAE, nous avons rencontré plusieurs problèmes :
- Le premier problème a été de relier le traitement des demandes d'ami sur le serveur avec l'affichage de ces demandes d'ami. En effet, dans les logs, la demande d'ami fonctionnait, mais pas en graphique. Pour y remédier, nous nous sommes documentés sur Internet et avons testé différentes solutions, jusqu'à ce que cela fonctionne.
- Le deuxième problème rencontré fut d'afficher les messages reçu. Nous avons également utilisé Internet afin de se sortir de cette situation.

Cette SAE nous à permis de travailler en totale autonomie et nous a permis de faire nous mêmes des recherches sur Internet afin de trouver des solutions au lieu d'appeler le professeur à chaque problèmes.