INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Rapport de UML et PDLA

Réalisé par

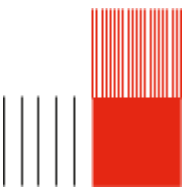
ZHANG Yu

Zeng yongjia

Groupe B1

INSA | INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Sommaire



Introduction

La réalisation de notre projet de chat système a été guidée par des pratiques de gestion de projet agiles et l'utilisation judicieuse d'outils tels que Jira et Maven.(Les itérations régulières et les réunions de sprint ont été orchestrées grâce aux fonctionnalités de planification de Jira, offrant une visibilité claire sur l'état d'avancement du projet.)

Jira

L'application de la méthode agile a été fondamentale dans la conduite de notre projet de chat par Jira. Cette approche nous a permis d'adopter une démarche itérative et incrémentale, garantissant une adaptation continue aux besoins tout en gardant une structure solide et évolutive pour notre application.

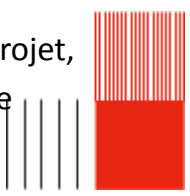
Sprint 1 : Structure de l'Application

Ce premier sprint a été une étape fondamentale dans la construction de notre application de chat.
Au cours de cette phase, nous avons tout d'abord développé les protocoles UDP et TCP, pour établir des connexions réseau qui c'est le principale de notre application. En même temps, la conception et l'implémentation de l'interface utilisateur graphique (GUI) est aussi dans ce sprint afin de garantir la convivialité de l'utilisateur. En parallèle, la connexion à la base de données a été configurée pour assurer un stockage des données nécessaires.

Sprint 2 : Interactions Utilisateurs

Le deuxième sprint a été consacré à l'amélioration des interactions directes entre les utilisateurs et notre application, introduisant des fonctionnalités cruciales pour une expérience utilisateur enrichie.
Les utilisateurs peuvent désormais se connecter et se déconnecter facilement, changer leur pseudonyme et consulter l'historique complet de leurs conversations. De plus,l'envoi et la réception de messages ont été mis en œuvre avec succès.

Grâce à cet outil, nous avons pu instaurer un processus dynamique de suivi du projet, offrant une visibilité cristalline sur l'évolution de chaque étape. Cela a assuré une



gestion agile et proactive du développement, renforçant ainsi notre capacité à maintenir un flux de travail efficace et adaptatif.

Maven

Maven a été utilisé dans la gestion de notre projet. Il a facilité la gestion des dépendances et la construction de développement.

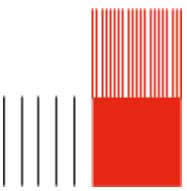
- Gestion des Dépendances : Toutes les bibliothèques nécessaires ont été définies dans le fichier pom.xml, garantissant la cohérence et la reproductibilité des environnements de développement et de production.
- Automatisation des Builds : Grâce à Maven, chaque push dans le repository déclenche un processus de build automatisé, s'assurant que toutes les intégrations étaient testées et construites systématiquement.

Github

Notre collaboration a été réalisée via une seule branche sur GitHub, pour simplifier notre gestion de code pour notre équipe de deux. Nous avons renforcé notre flux de travail avec GitHub Actions pour l'intégration continue.
À chaque commit, Git Actions déclenche :

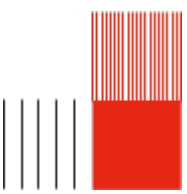
- Un build Maven automatique, assurant la compilation et l'empaquetage du code.
- L'exécution de tous les tests, confirmant la fiabilité de chaque changement.

Cette configuration a assuré que notre application était testée, permettant une intégration rapide et sûre de nouvelles fonctionnalités.



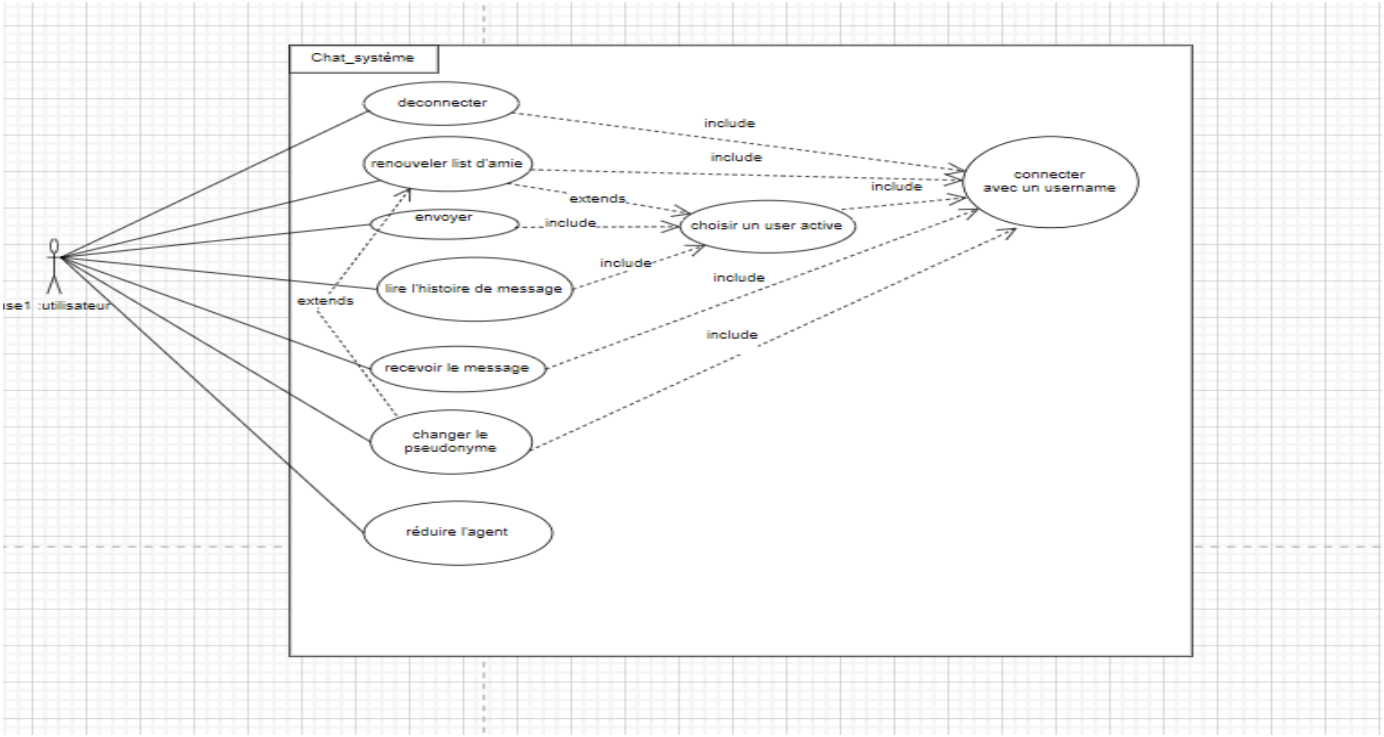
Conclusion

Grâce à l'intégration réussie de Jira et Maven, les deux sprints ont été livrés dans les délais prévus. Toutes les fonctionnalités spécifiées ont été implémentées avec succès, jetant les bases solides d'une application de chat système fiable et fonctionnelle.
Ces outils ont fourni une visibilité et un contrôle accrus sur le développement, permettant une gestion efficace des tâches et une automatisation fiable du cycle de vie du projet. Nous sommes optimistes quant au résultat de notre application de chat, construite sur une base solide grâce à ces outils de gestion et d'automatisation.

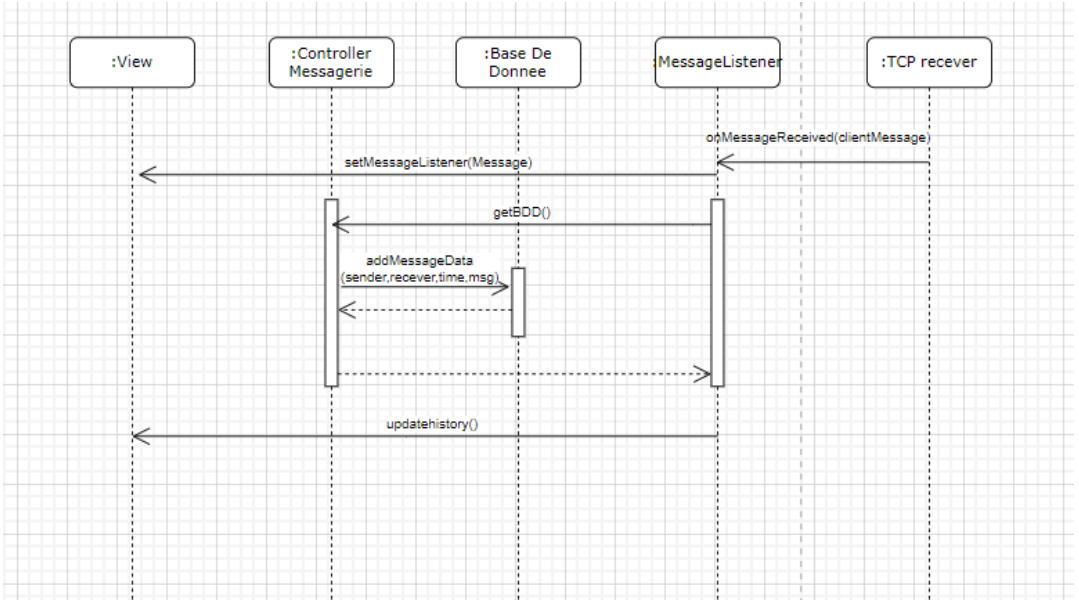


Annexe

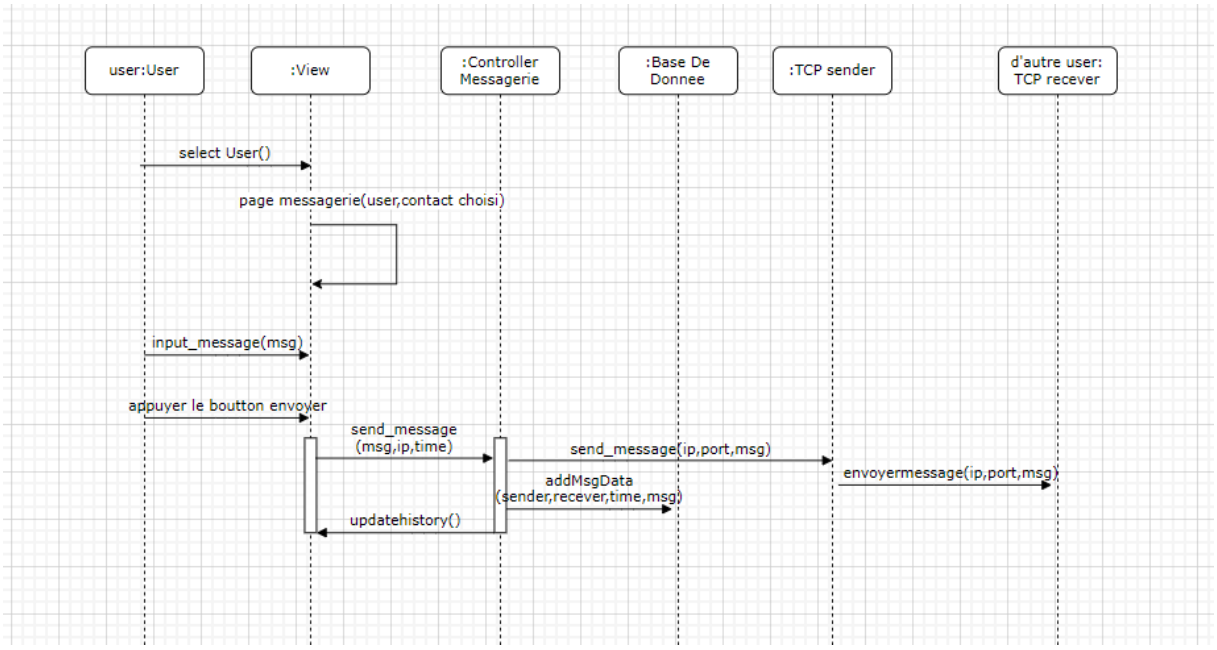
USE CASE :



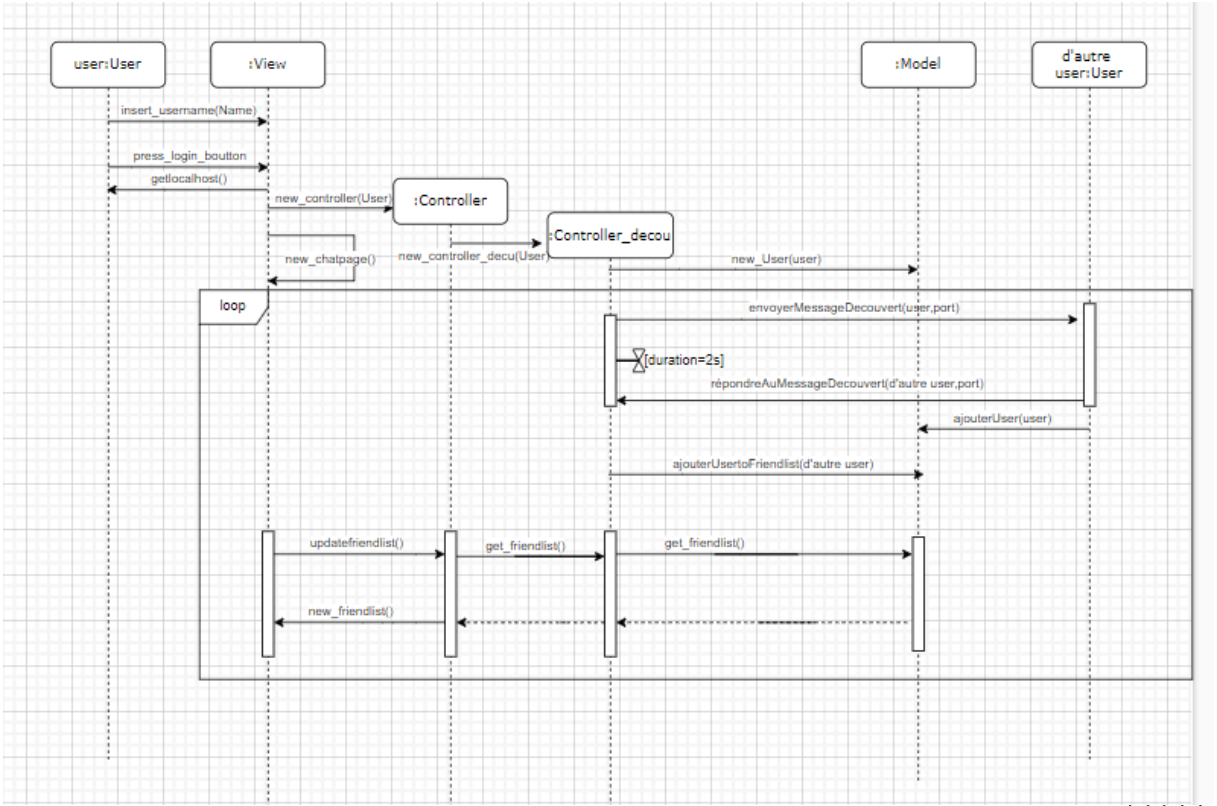
SÉQUENCE ENVOYER:



SÉQUENCE RECEIVER:



SÉQUENCE CONNECTION:



SÉQUENCE DÉCONNECTION:

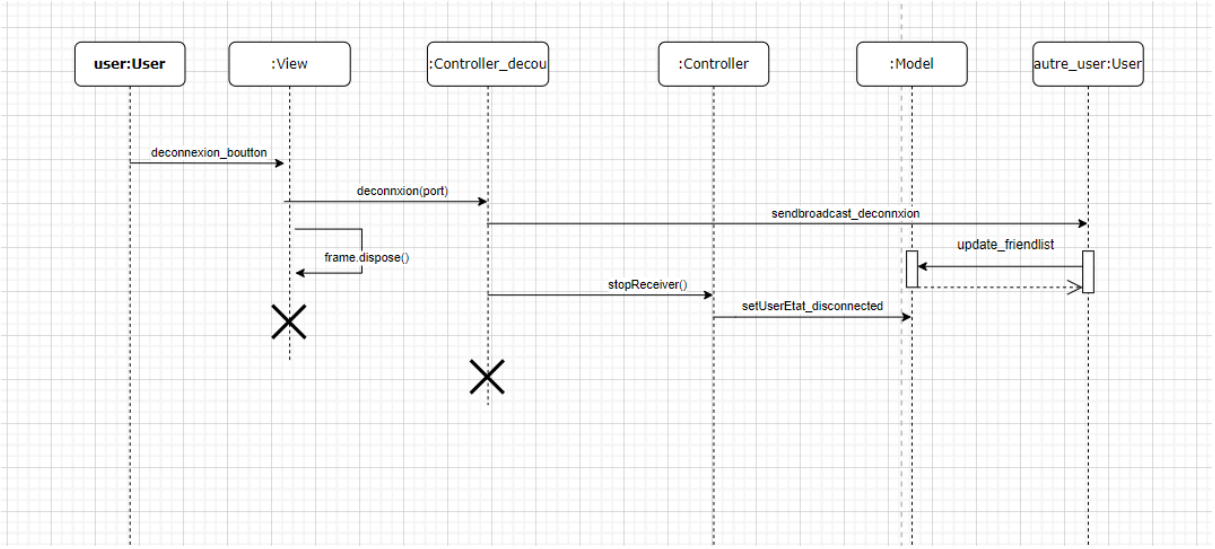
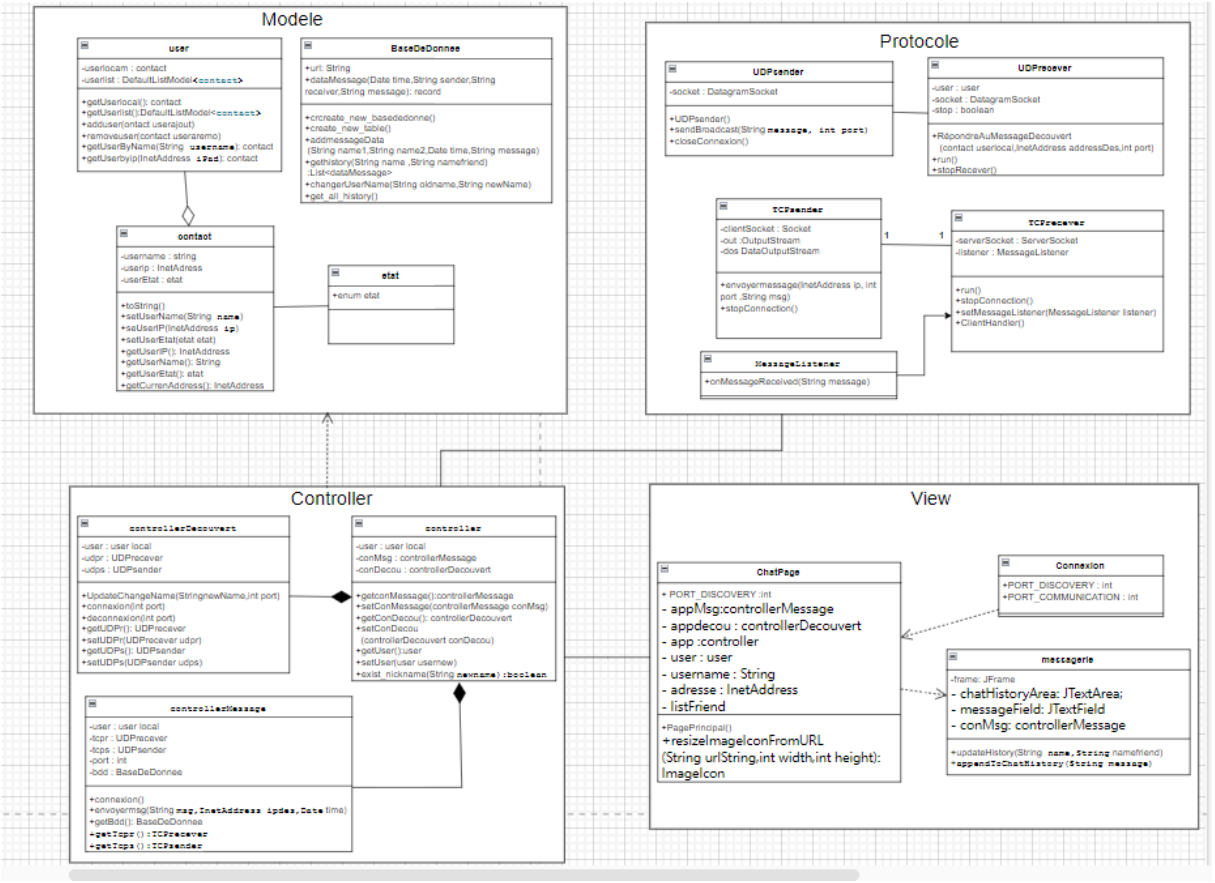
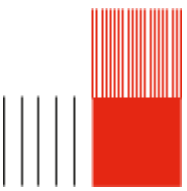
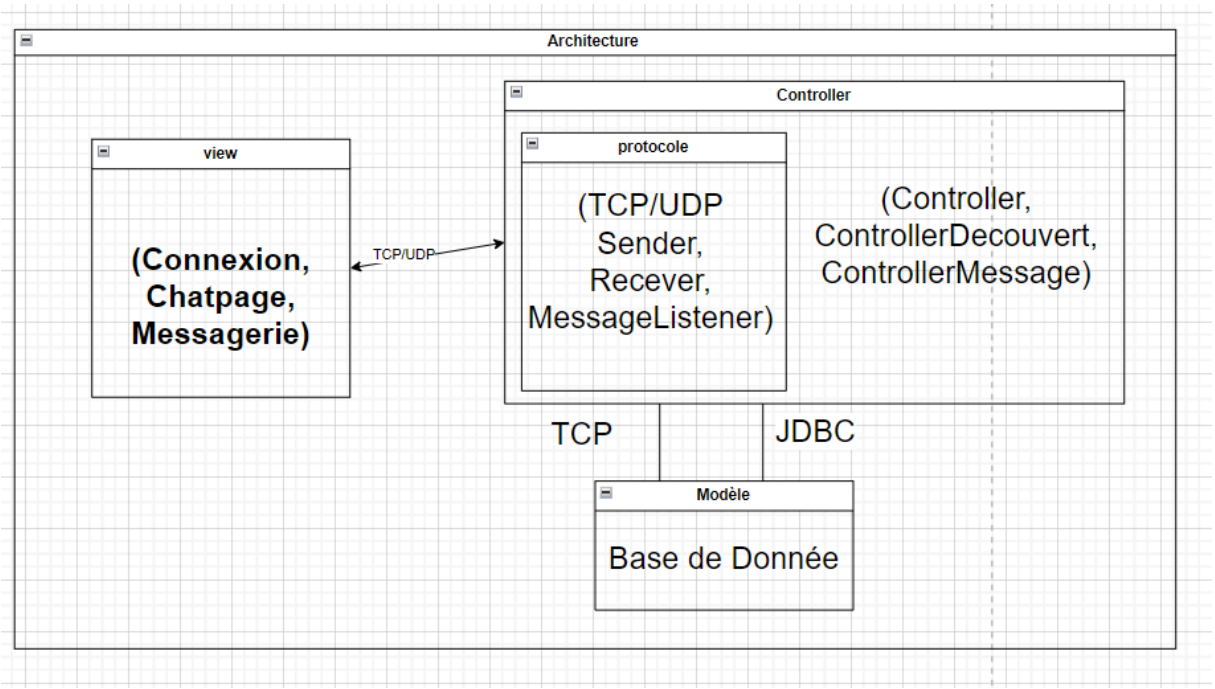


DIAGRAMME DE CLASSE:

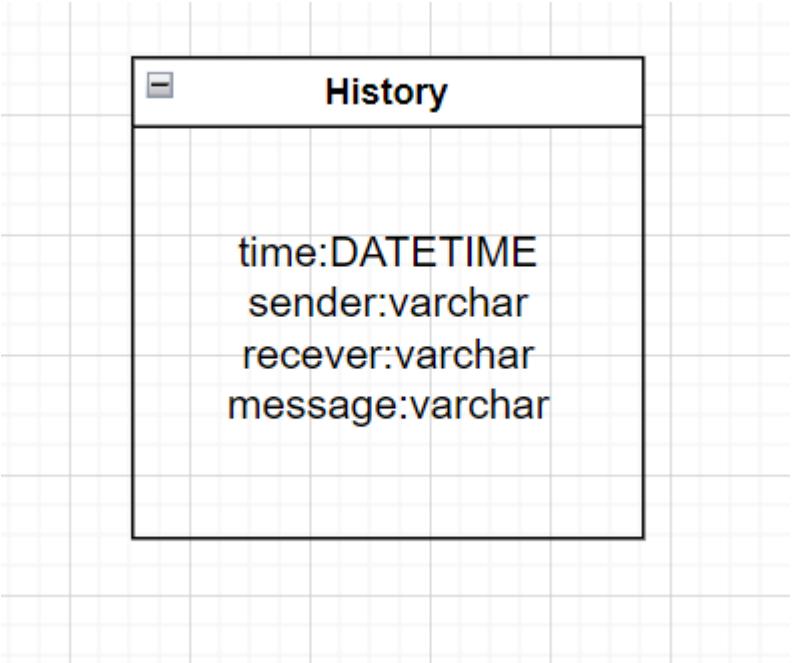


ARCHITECTURE:





DATABASE:



pour base de données, on n’a qu' une tableau qui contient tous les messages de user local, et chaque fois , envoyer le message ou recevoir le message, il enregistre dedans , et la sélection par sender et receiver.