UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE

Système de QCM avec RabbitMQ

Compte rendu TP4 INFO502

Élève :
Moussa TAYEB NEMICHE

Enseignants :
OLIVIER FLAUZAC
GEOFFREY WILHELM

Table des matières

1 Introduction

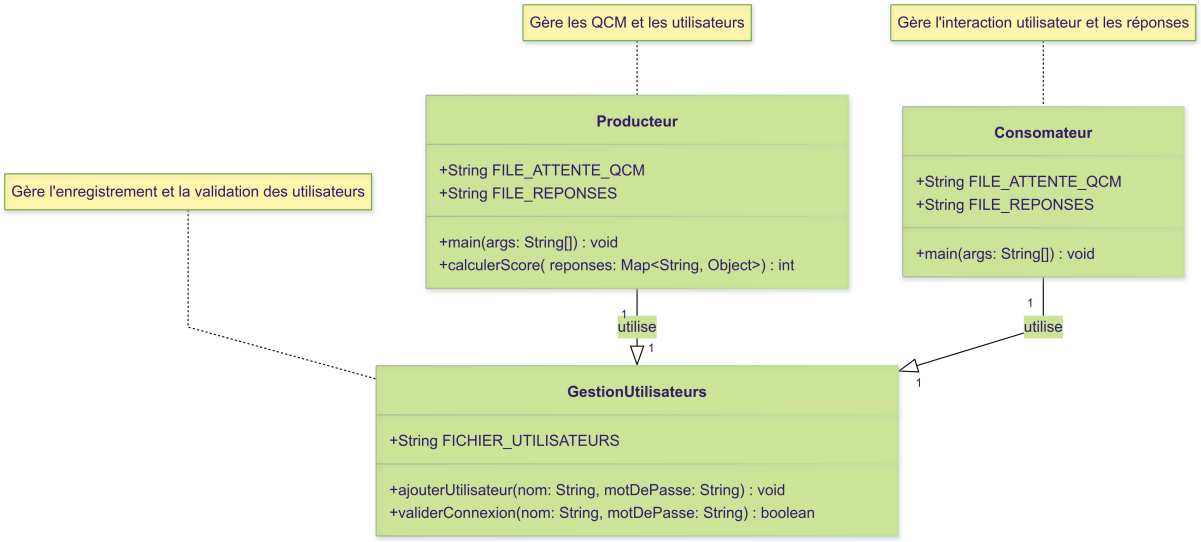
Dans ce rapport, je vais vous présenter un système de gestion de questionnaires à choix multiples (QCM) qui utilise RabbitMQ pour la communication entre un serveur et un client. Ce système permet non seulement de gérer les QCM, mais aussi d'enregistrer les utilisateurs et de vérifier leurs informations de connexion. Je vais expliquer l'architecture du système, les différents composants du code, et les fichiers de messages utilisés.

2 Architecture du Système

L'architecture de ce système repose sur un modèle client-serveur, avec RabbitMQ comme intermédiaire pour la communication asynchrone. Voici les composants principaux :

- **Producteur (Serveur)** : Il gère les QCM, enregistre les utilisateurs et vérifie leurs informations de connexion.
- **Consomateur (Client)** : Il permet aux utilisateurs de s'enregistrer, de se connecter, de recevoir des QCM, de répondre aux questions et de recevoir leurs scores.
- **RabbitMQ** : C'est un broker de messages qui implémente la fonctionalité du producteur/consomateur.

Le serveur lit un fichier JSON contenant le QCM et l'envoie à la file d'attente RabbitMQ. Le consomateur, de son côté, écoute cette file d'attente pour recevoir le QCM, affiche les questions à l'utilisateur, collecte les réponses et les envoie au serveur via une autre file d'attente. Le producteur calcule ensuite le score et l'envoie au client.



3 Composants du Code

3.1 Producteur (Serveur)

Le serveur a plusieurs responsabilités : envoyer les QCM aux clients, recevoir les réponses, calculer les scores et gérer les utilisateurs.

3.1.1 Envoi du QCM

Le serveur commence par lire un fichier JSON qui contient le QCM. Ce fichier inclut les questions, les choix de réponses et les réponses correctes. Ensuite, le serveur utilise la bibliothèque RabbitMQ pour publier le QCM dans une file d'attente spécifique que le client écoute.

3.1.2 Réception des Réponses et Calcul du Score

Une fois que le producteur a envoyé le QCM, il écoute la file d'attente des réponses pour recevoir les réponses des clients. Lorsqu'il reçoit les réponses, il les compare avec les réponses correctes du QCM pour calculer le score. Ce score est ensuite envoyé au consomateur via une autre file d'attente RabbitMQ.

3.1.3 Gestion des Utilisateurs

Le serveur gère également l'enregistrement et la connexion des utilisateurs en utilisant un fichier JSON (users.json). Lors de l'enregistrement, les informations de l'utilisateur (nom d'utilisateur et mot de passe) sont ajoutées au fichier. Lors de la connexion, le serveur vérifie les informations d'identification de l'utilisateur en comparant les données entrées avec celles stockées dans le fichier.

3.2 Consomateur (Client)

Le consomateur peut s'enregistrer, de se connecter, de recevoir des QCM, de répondre aux questions et de recevoir leurs scores.

3.2.1 Enregistrement et Connexion

Le consomateur propose un menu permettant aux utilisateurs de s'enregistrer ou de se connecter. Lors de l'enregistrement, les informations de l'utilisateur sont envoyées au serveur pour être stockées dans le fichier users.json. Lors de la connexion, les informations d'identification de l'utilisateur sont vérifiées par le serveur.

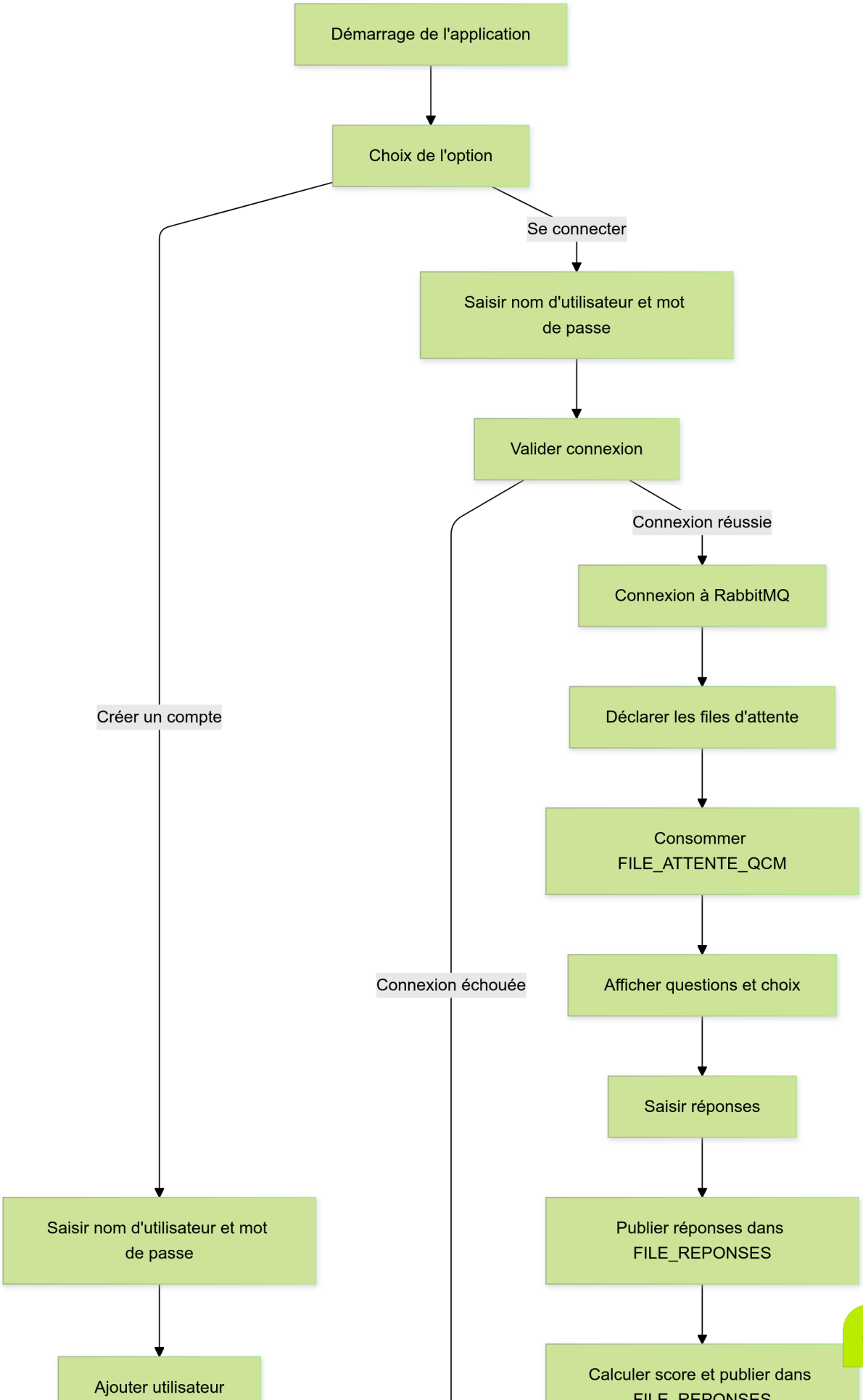
3.2.2 Réception du QCM et Envoi des Réponses

Le consomateur écoute la file d'attente des QCM pour recevoir le QCM envoyé par le serveur. Une fois le QCM reçu, le client affiche les questions à l'utilisateur, collecte les

réponses et les envoie au serveur via une file d'attente RabbitMQ. Le client utilise un scanner pour lire les réponses de l'utilisateur et les envoie au serveur sous forme de JSON.

3.2.3 Réception du Score

Le client écoute également la file d'attente des scores pour recevoir le score envoyé par le serveur. Une fois le score reçu, le client l'affiche à l'utilisateur.


Démarrage de l'application
Choix de l'option
Se connecter
Saisir nom d'utilisateur et mot de passe
Valider connexion
Connexion réussie
Connexion à RabbitMQ
Déclarer les files d'attente
Consommer FILE_ATTENTE_QCM
Afficher questions et choix
Saisir réponses
Publier réponses dans FILE_REPONSES
Calculer score et publier dans
Connexion échouée
Créer un compte
Saisir nom d'utilisateur et mot de passe
Ajouter utilisateur

4 Fichiers de Messages

Les fichiers de messages utilisés dans ce système sont les suivants :

- **qcm.json** : Ce fichier contient les questions et les réponses du QCM. Chaque question a un identifiant unique, un texte, des choix de réponses et une réponse correcte.
- **users.json** : Ce fichier contient les informations des utilisateurs enregistrés. Chaque utilisateur a un nom d'utilisateur et un mot de passe.

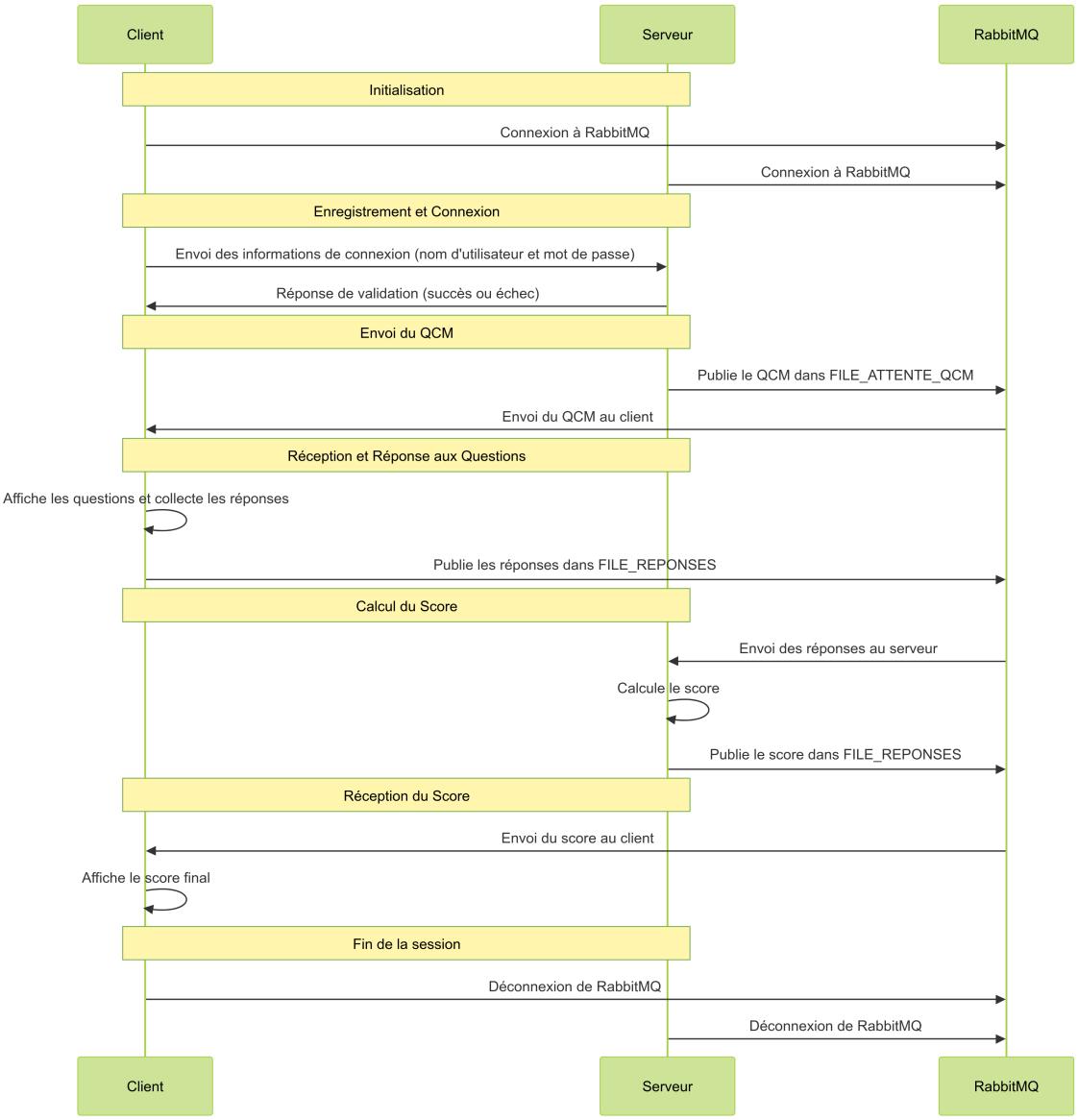
Ces fichiers JSON sont utilisés pour stocker les données de manière structurée et facilement lisible par les composants du système.

5 Choix de l'Architecture et des Types de Données

5.1 Choix de l'Architecture

L'architecture producteur/consomateur avec RabbitMQ a été choisie pour plusieurs raisons :

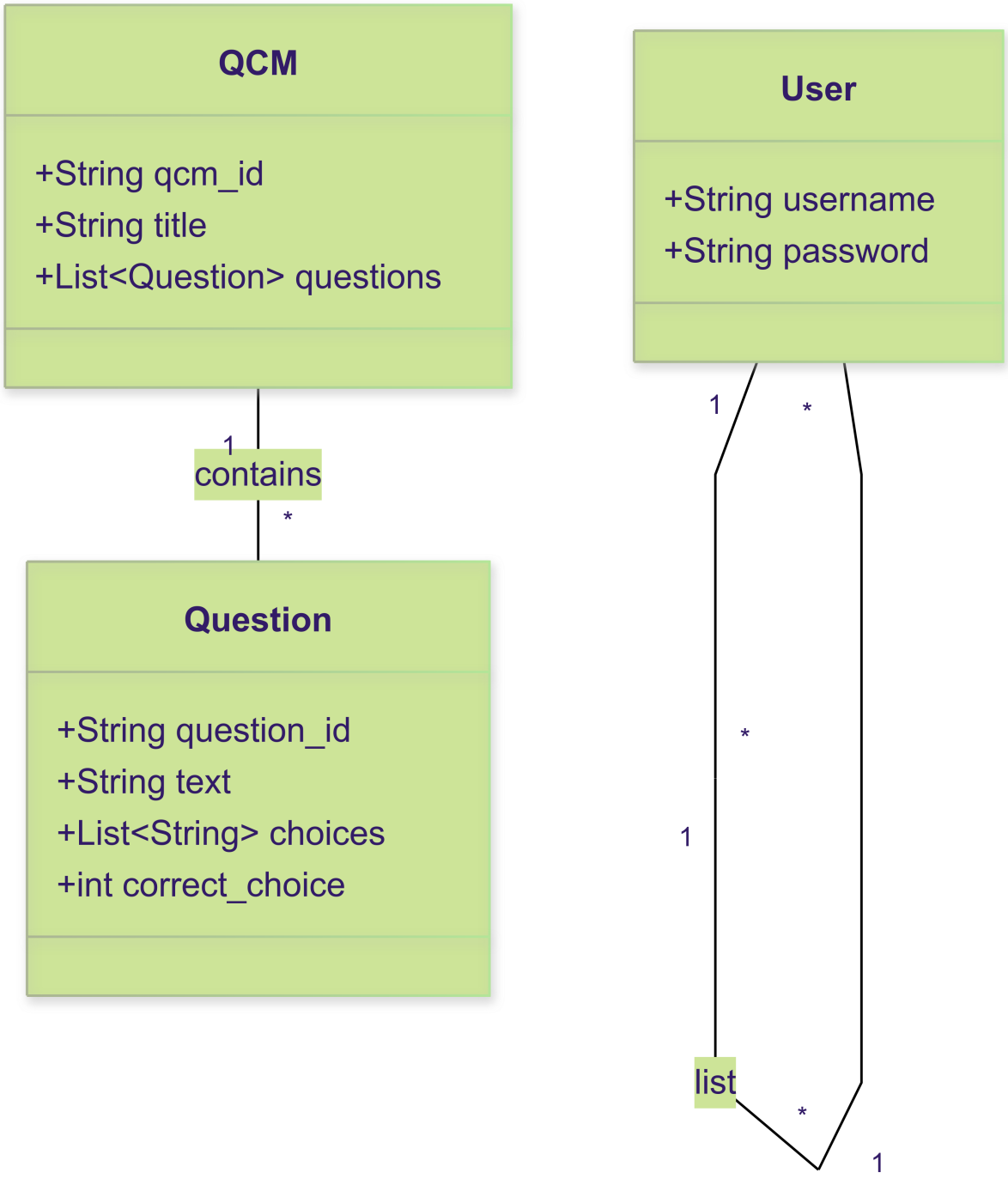
- **Séparation de rôles** : Le producteur et le consomateur sont découplés, ce qui permet de développer et de déployer chaque composant indépendamment.
- **Scalabilité** : RabbitMQ permet de gérer un grand nombre de clients et de messages de manière efficace.
- **Fiabilité** : RabbitMQ assure la livraison des messages même en cas de défaillance temporaire du serveur ou du client.



5.2 Types de Données

Les types de données utilisés dans ce système sont principalement des objets JSON pour les QCM et les utilisateurs. Voici pourquoi ces types de données ont été choisis :

- **JSON** : Le format JSON est léger, facile à lire et à écrire, et largement supporté par de nombreuses bibliothèques et langages de programmation.
- **Listes et Maps** : Les listes et les maps en Java permettent de manipuler facilement les données JSON et de les convertir en objets Java pour un traitement plus facile.



6 Conclusion

Dans ce rapport, j'ai présenté un système de gestion de QCM utilisant RabbitMQ pour la communication entre un producteur et un consomateur. Ce système permet non seulement de gérer les QCM, mais aussi d'enregistrer les utilisateurs et de vérifier leurs informations

de connexion. J'ai expliqué l'architecture du système, les différents composants du code, et les fichiers de messages utilisés. Ce système permet une communication asynchrone efficace et une gestion centralisée des QCM et des utilisateurs.