

Projet BitRuisseau

BitRuisseau

Mateen Salem Khalil
08/11/2024

Table des matières

Introduction	2
Analyse.....	2
Journal de travail	4
Etat des lieux	5
Démonstration du programme	6
Utilisation des fonctionnalités.....	Erreur ! Signet non défini.
Journal de travail	9
Conclusion	9
ChatGPT	10

Introduction

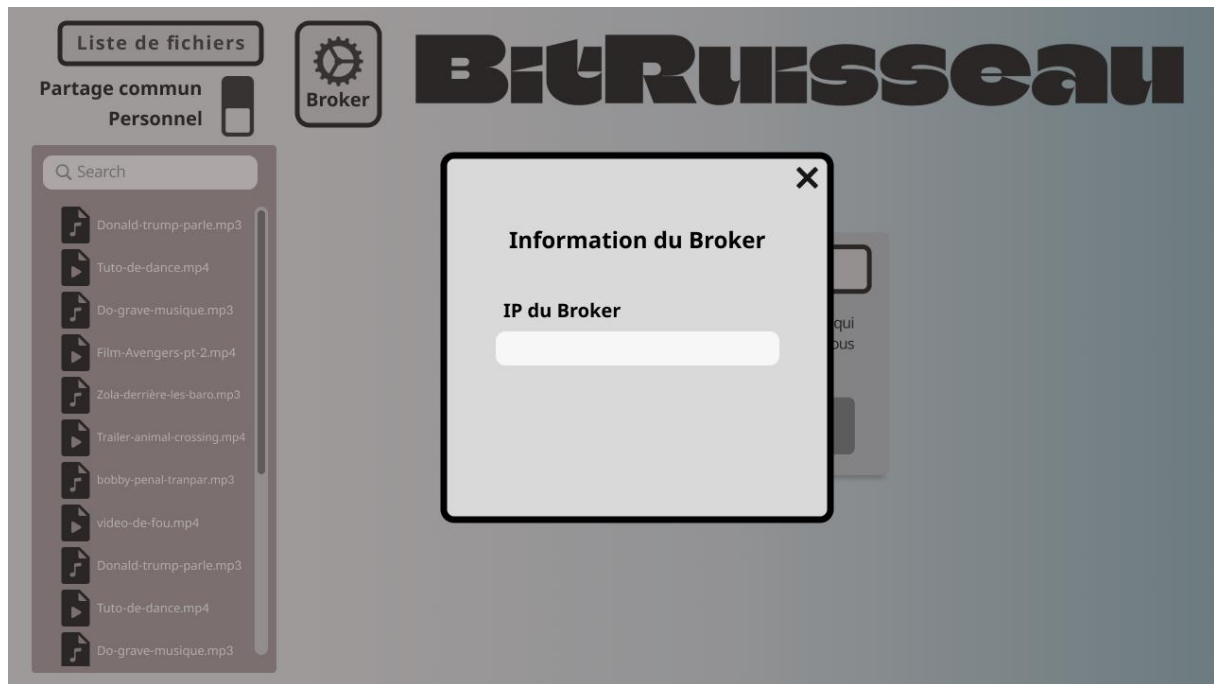
Dans le cadre de ce projet, nous avons développé une application mettant en œuvre les principes fondamentaux de la programmation distribuée, avec un accent particulier sur la communication inter-applications via le protocole MQTT. L'objectif principal était de créer un système capable de gérer l'échange d'informations concernant des catalogues multimédias, notamment des musiques, en facilitant à la fois le partage des métadonnées et des fichiers binaires correspondants.

Pour réaliser cet objectif, une architecture de communication normée a été mise en place. Cette architecture repose sur la définition de divers types de messages, chacun conçu pour répondre à des cas d'usage précis tels que la demande ou l'envoi de catalogues, ainsi que le transfert de fichiers. Les messages sont structurés de manière à garantir leur compréhension et leur efficacité dans un environnement réparti.

Analyse



Maquette 1



Maquette 2

Section	Élément	Fonctionnalité
Navigation	Liste de fichiers	Permet d'afficher la liste de tous les fichiers disponibles pour lecture et partage.
	Toggle partage commun/Personnel	Permet de basculer entre les fichiers partagés publiquement et les fichiers personnels.
	Broker (Réglages)	Ouvre un pop-up pour configurer l'adresse IP, port, username, password du broker MQTT.
	Barre de recherche	Permet de rechercher un fichier spécifique dans la liste de fichiers. (Spécifier comment elle fonctionne !!!!)
Gestion des Fichiers	Liste des fichiers	Affiche les fichiers disponibles sous forme de liste avec des icônes pour différencier les types (audio, vidéo). (Attributs des médias : type du fichier exemple : mp3, mp4, etc. Nom du fichier)
	Scrolling	Permet de naviguer entre les fichiers en défilant la liste.

	Double-clic fichier - Partage commun	Lorsqu'un fichier en partage commun est double-cliqué, il se télécharge dans le dossier localisé par l'utilisateur. Si aucun dossier n'est localisé, un pop-up s'affiche pour informer l'utilisateur qu'il doit d'abord sélectionner un dossier de stockage pour télécharger des fichiers partagés.
	Double-clic fichier - Local	Lorsqu'un fichier local est double-cliqué, il s'ouvre automatiquement pour lecture. Si le fichier est un fichier audio (ex. : mp3), il sera joué.
Dossier de Médias	Titre et description	Indique à l'utilisateur la fonction du dossier (contenant les fichiers à partager publiquement).
	Bouton « Localiser »	Permet de sélectionner le dossier qui contient les fichiers à partager avec les autres utilisateurs. Si dossier change, les médias partager ne seront plus indexer (fenêtre de confirmation quand on clique sur le bouton « Localiser » alors qu'un dossier l'est déjà
Configuration du Broker	Pop-up info du Broker	Affiche une fenêtre pour saisir l'adresse IP, port, username, password du broker MQTT, permettant la connexion pour le partage de fichiers.
	Fenêtre information "Information du Broker"	Permet à l'utilisateur de saisir l'adresse IP, port, username, password du broker MQTT pour établir la connexion nécessaire au partage et réception des fichiers.

Journal de travail

Etat des lieux

1. Fonctionnalités Complètes :

Localisation des Médias Locaux : L'application permet de localiser un dossier contenant les fichiers multimédias locaux et d'afficher la liste des fichiers disponibles.

Modification de la Configuration du Broker : Les utilisateurs peuvent modifier les paramètres de leur broker afin de se connecter aux autres utilisateurs.

Envoi et Récupération des Catalogues : L'utilisateur peut envoyer son catalogue de fichiers à partager et récupérer les catalogues des autres utilisateurs, facilitant ainsi la gestion des fichiers partagés.

Affichage des Médias Locaux et Partagés : Les fichiers locaux sont listés correctement, et les fichiers partagés dans les catalogues sont également affichés à l'utilisateur, offrant une vue d'ensemble des ressources disponibles.

2. Fonctionnalités Partiellement Fonctionnelles :

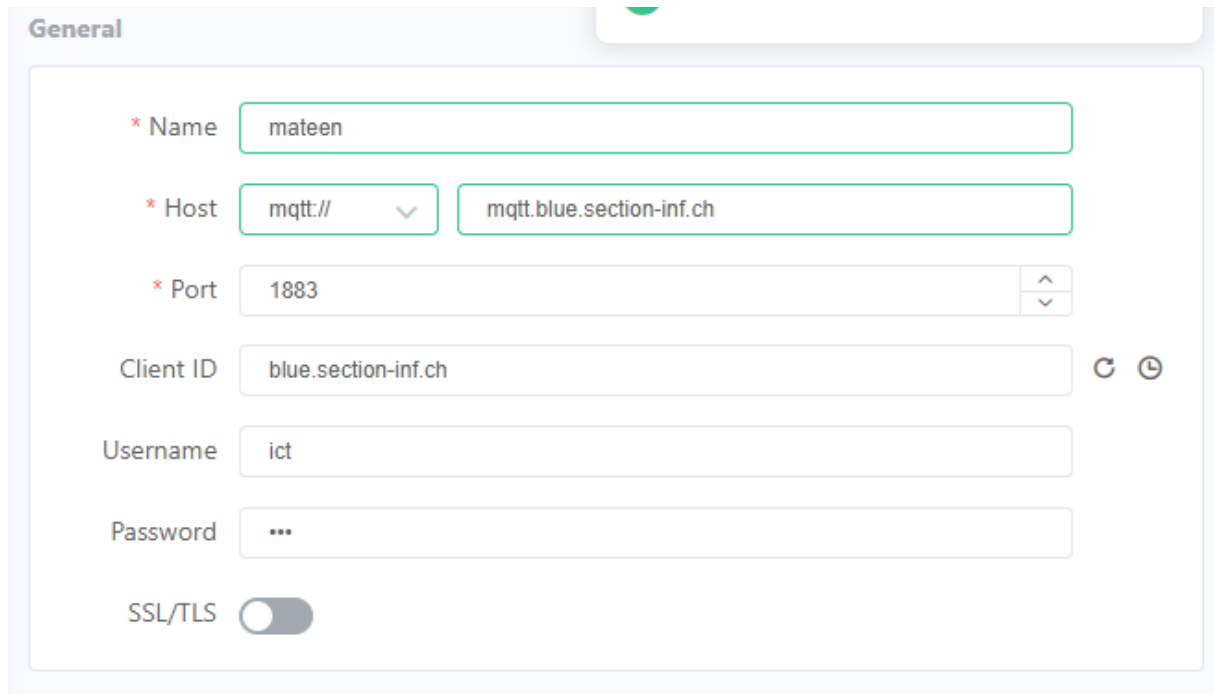
Demande de Téléchargement de Fichiers : L'utilisateur peut demander à télécharger un fichier à un autre utilisateur via le broker. Cependant, des erreurs techniques empêchent cette fonctionnalité d'être pleinement fonctionnelle. La demande de téléchargement échoue dans certains cas, et la logique de traitement des demandes reste instable.

Envoi de Fichiers Demandés : Bien que l'utilisateur puisse envoyer un fichier demandé par un autre utilisateur, cette fonctionnalité rencontre également des erreurs, notamment au niveau de la transmission des fichiers. Ces erreurs rendent l'envoi de fichiers partagés non fiable à ce stade.

Démonstration du programme

Configuration Mqtt :

Voici la configuration mqtt que j'ai utilisé pour mon programme



The screenshot shows a 'General' tab for MQTT configuration. The fields are as follows:

- Name:** mateen
- Host:** mqtt:// (dropdown) and mqtt.blue.section-inf.ch
- Port:** 1883 (with up/down arrows)
- Client ID:** blue.section-inf.ch (with refresh and save icons)
- Username:** ict
- Password:** ... (masked)
- SSL/TLS:** Toggle switch is turned off.

Mon programme utilise par défaut le topic « test »

Scénario de test du programme

1. Envoie de catalogue :

Objectif : Tester si le programme envoie correctement son catalogue à la demande.

Étapes :

1. L'utilisateur demande le catalogue via un émetteur ou en localisant le dossier des médias.
2. Le programme envoie le catalogue à l'émetteur.

Résultat attendu : Le programme doit envoyer le catalogue de manière adéquate sans erreurs

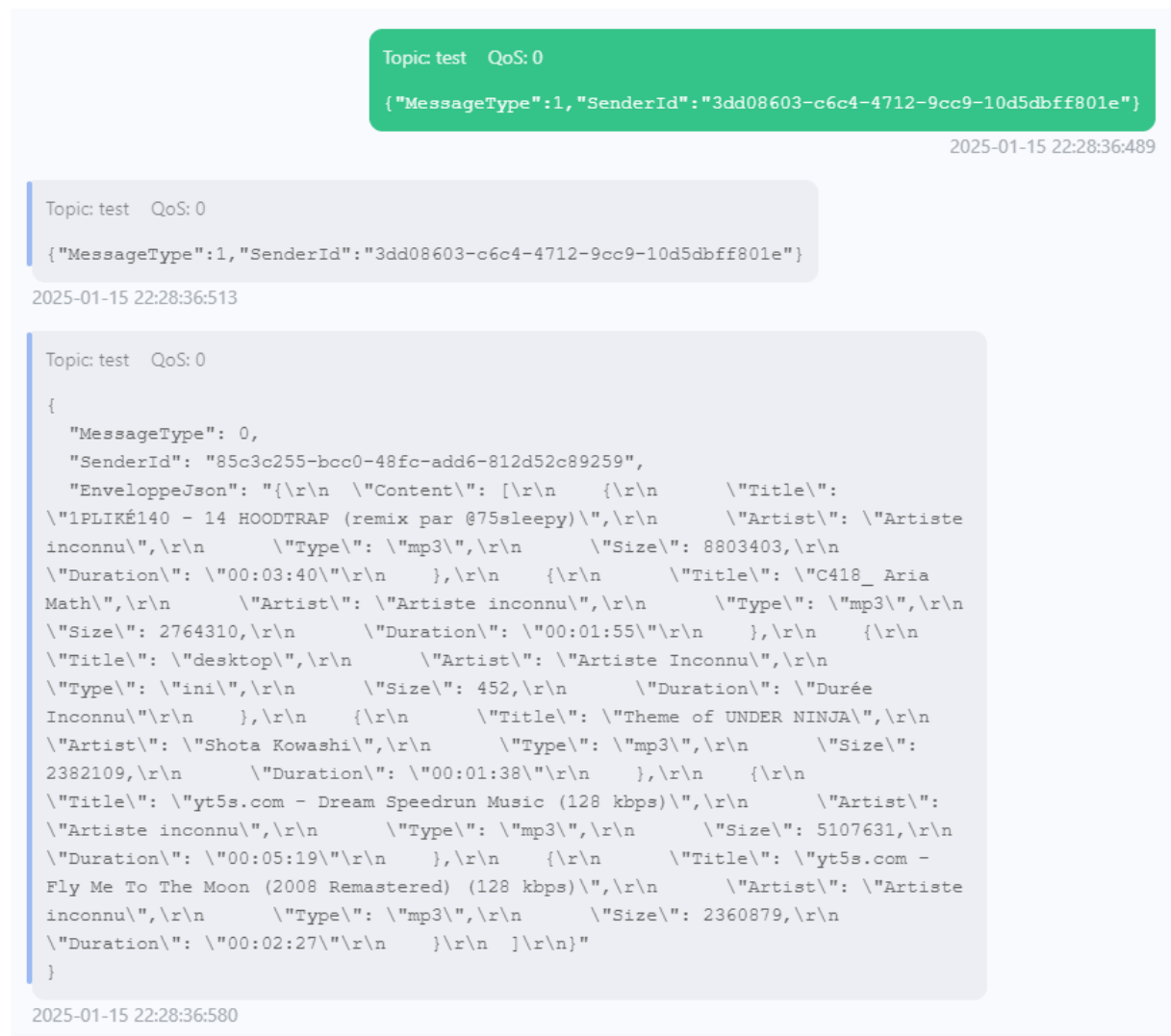


Figure 1 envoi de catalogue après une demande

2. Réception de catalogue :

Objectif : Vérifier si le programme ajoute un média dans la liste des fichiers reçus lors de la réception d'un nouveau catalogue.

Étapes :

1. Un catalogue est reçu par le programme.
2. Le programme ajoute le média à la liste des fichiers locaux.

Résultat attendu : Le média est ajouté à la liste des fichiers sans erreurs.



Figure 2 musique reçu et ajouté dans le programme

3. Demande de fichier :

Objectif : Tester si l'utilisateur peut demander un fichier en cliquant simplement sur le fichier qu'il souhaite télécharger.

Étapes :

1. L'utilisateur sélectionne un fichier à télécharger en cliquant dessus.
2. Le programme envoie la demande de téléchargement du fichier.

Résultat attendu : Le fichier est correctement demandé pour téléchargement sans erreurs.



Figure 3 Demande de fichier envoyé par le programme

4. Envoie de fichier :

Objectif : Vérifier le comportement du programme lorsqu'il reçoit une demande d'envoi de fichier.

Étapes :

1. Le programme reçoit une demande d'envoi de fichier.
2. Le programme tente d'envoyer le fichier demandé.

Cela a été un défi de suivre leur rythme, surtout lorsque certaines décisions avaient été prises sans que tout le monde soit au courant ou aligné.

Cela dit, malgré ces difficultés, le projet a été une excellente opportunité d'apprentissage. L'expérience de travailler avec MQTT et de réaliser une analyse technique a été très enrichissante. J'ai aussi apprécié les aspects pratiques du projet, mais je pense que des améliorations dans la gestion du travail d'équipe et la définition des protocoles communs rendraient cette expérience encore plus fluide pour tout le monde.

ChatGPT

L'IA m'a été utile tout au long du projet, notamment pour comprendre la logique des codes de mes camarades et rendre leurs méthodes compatibles avec mon propre code. Elle m'a expliqué des segments de code complexes et m'a permis de mieux m'intégrer au projet en adaptant des méthodes pour qu'elles fonctionnent avec mon architecture. L'IA m'a aussi aidé à résoudre des erreurs en identifiant leurs causes et en suggérant des solutions, tout en m'offrant des conseils sur la gestion de la communication MQTT et la programmation distribuée.