

Rapport P_BitRuisseau

Julien Mares

Introduction	2
Maquette & Utilisations	2
Page MediaPlayer	2
Page Mediatheque	3
Analyse fonctionnelle	4
Planification	6
Etat des lieux	6
Démonstration d'utilisation (solo).....	7
Configuration Mqttx	7
Scénario de test du programme	8
1. Demande de catalogue.....	8
2. Reception catalogue	8
3. Demande de musique	9
4. Réception musique	9
A lire concernant les tests	10
Arborescence test solo	10
Modification SenderId	10
Clarifications des différents scénarios de test	11
Utilisation AI dans ce projet	12
Conclusion	12
Journal de travail	12

Introduction

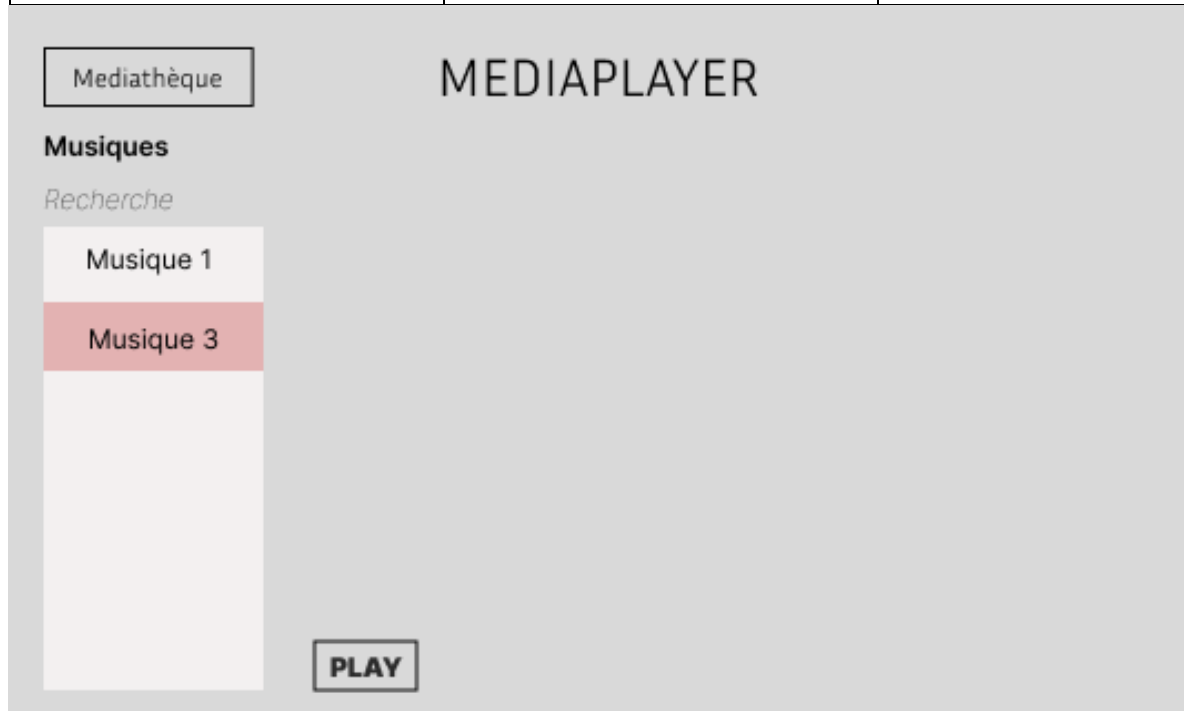
Dans ce projet, nous allons concevoir un programme illustrant les principes de la programmation distribuée en mettant en place une communication entre applications via le protocole MQTT. Ce système permettra d'échanger des informations sur des catalogues de musiques, allant de l'envoi d'un catalogue et des informations relatives aux musiques jusqu'au partage de fichiers binaires.

Pour ce faire, nous définirons un système de communication normé, avec plusieurs types de messages adaptés à chaque cas d'usage (demande de catalogues, envoi de catalogues, demande de fichiers, envoi de fichiers). Une architecture générale de message sera établie, et la sérialisation des données sera utilisée pour transporter efficacement ces informations à travers le réseau MQTT.

Maquette & Utilisations

Page MediaPlayer

Elements	Fonctions	Divers
Titre « MEDIAPLAYER »	Spécifier sur quelle page on est	
Bouton « Mediathèque »	Dirige vers la page « Mediathèque »	
Liste des musiques/vidéo	Liste des musiques/vidéos disponibles localement	L'affichage de chaque musique/vidéo contient : son titre, son auteur, sa taille.
Placeholder « Recherche »	Champ de recherche pour la liste	
Bouton « PLAY »	Lire la musique/vidéo sélectionné dans la liste	



Page Mediatheque

Elements	Fonctions	Divers
Titre « MEDIAPLAYER »	Spécifier sur quelle page on est	
Bouton « MediaPlayer »	Dirige vers la page « MediaPlayer »	
Liste « Communauté »	Liste-les musiques/vidéos disponibles sur le réseau	L'affichage de chaque musique/vidéo contient : son titre, son auteur, sa durée.
Liste « Fichier locaux »	Liste-les musiques/vidéos téléchargé localement	L'affichage de chaque musique/vidéo contient : son titre, son auteur, sa durée.
Boutons « dl »	Permet de télécharger des musiques/vidéos du réseau localement	
Placeholders « Recherche »	Trier les musiques/vidéos de sa liste respective	
Icone de config (en haut à droite)	Permet de configurer le broker	On le configure avec : IP broker Port username password
Icone d'Upload (en bas de la liste « Fichier locaux »)	Permet d'ajouter une musiques/vidéos à la médiathèque via des fichiers locaux	



Analyse fonctionnelle

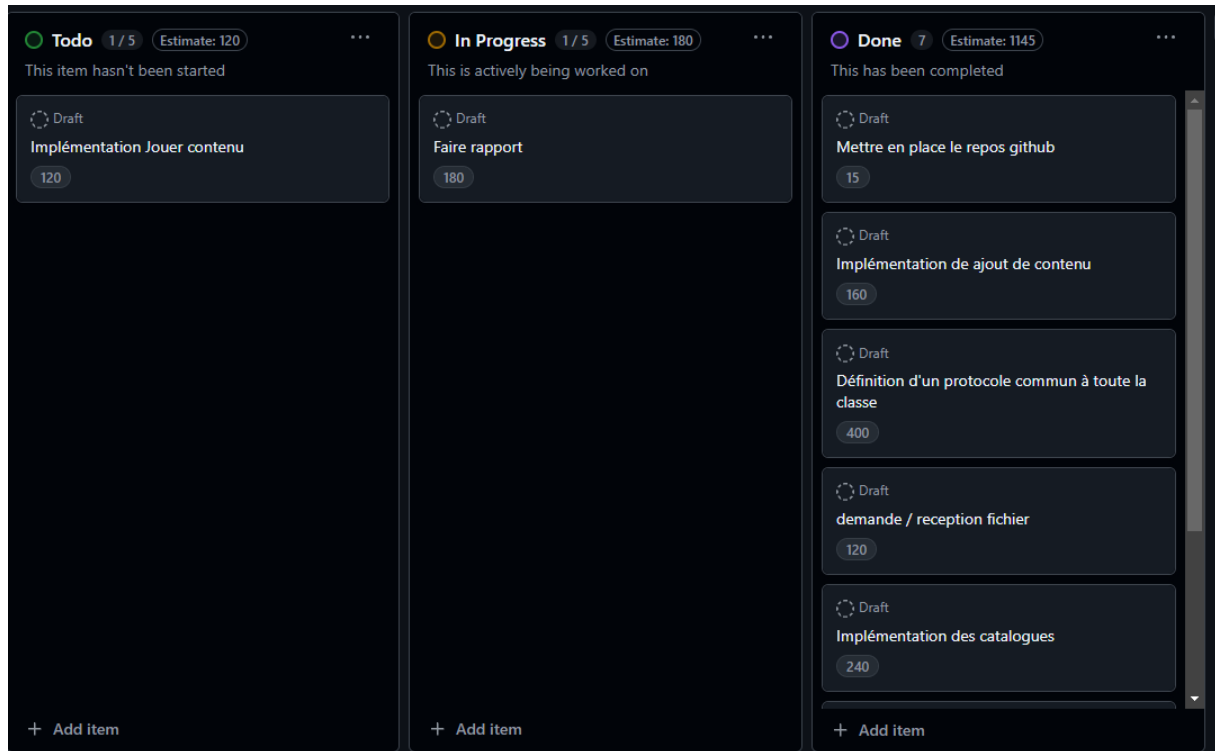
Fonctionnalité	Description	Tests d'acceptance
Liste musiques/vidéos	J'ai besoin d'une liste des musiques/vidéos pour choisir quoi écouter/regarder.	<p>- Sélection d'un objet :</p> <ol style="list-style-type: none"> 1. Je clique sur un élément de la liste. 2. L'arrière-plan de l'élément devient coloré. 3. Tout autre élément sélectionné précédemment perd son état sélectionné. <p>- Aucun objet sélectionné :</p> <ol style="list-style-type: none"> 1. Je clique sur "PLAY". 2. Un message s'affiche : "Veuillez sélectionner un fichier pour continuer."
Bouton Play	Je veux un bouton "PLAY" pour lancer la musique/vidéo sélectionnée.	<p>- Lecture d'un fichier valide :</p> <ol style="list-style-type: none"> 1. Je sélectionne un fichier valide dans la liste. 2. Je clique sur "PLAY". 3. La musique/vidéo démarre. <p>- Lecture d'un fichier non supporté :</p> <ol style="list-style-type: none"> 1. Je sélectionne un fichier au format non pris en charge. 2. Je clique sur "PLAY". 3. Un message d'erreur s'affiche : "Format non pris en charge." <p>- Arrêt de la lecture :</p> <ol style="list-style-type: none"> 1. Une musique/vidéo est en cours de lecture. 2. Je clique sur "STOP". 3. La lecture s'arrête immédiatement.
Bouton de navigation	J'ai besoin de boutons pour naviguer entre les pages de l'application.	<p>- Redirection vers Médiathèque :</p> <ol style="list-style-type: none"> 1. Je clique sur le bouton "Médiathèque". 2. Je suis redirigé vers la page Médiathèque. <p>- Retour vers MediaPlayer :</p> <ol style="list-style-type: none"> 1. Je clique sur le bouton "MediaPlayer". 2. Je retourne sur la page MediaPlayer.
Ajout de musiques	Je veux importer des musiques de mon PC pour les rendre accessibles via l'application.	<p>- Ajout d'un fichier valide :</p> <ol style="list-style-type: none"> 1. Je clique sur le logo "Upload". 2. L'explorateur de fichiers s'ouvre.

		<p>3. Je sélectionne un fichier au format supporté (.mp3/.mp4).</p> <p>4. Le fichier est ajouté à la bibliothèque et visible dans la liste.</p> <p>- Ajout d'un fichier non supporté :</p> <p>1. Je clique sur "Upload".</p> <p>2. Je sélectionne un fichier au format non supporté.</p> <p>3. Un message d'erreur s'affiche : "Format non pris en charge."</p> <p>- Ajout d'un fichier déjà présent :</p> <p>1. J'essaie d'ajouter un fichier déjà importé.</p> <p>2. Un message s'affiche : "Fichier déjà existant dans la bibliothèque."</p>
Configuration Broker	J'ai besoin de configurer un broker pour me connecter à différents serveurs.	<p>- Connexion valide :</p> <p>1. Je clique sur l'icône de configuration du broker.</p> <p>2. Une fenêtre s'ouvre.</p> <p>3. Je saisis les paramètres du broker (IP, port, identifiants).</p> <p>4. Je clique sur "Se connecter".</p> <p>5. La connexion est établie et la liste des musiques/vidéos se met à jour.</p> <p>- Connexion invalide :</p> <p>1. Je saisis des informations incorrectes dans la fenêtre de configuration.</p> <p>2. Je clique sur "Se connecter".</p> <p>3. Un message d'erreur s'affiche : "Connexion échouée. Vérifiez vos paramètres."</p>

Planification

Pour ce projet j'ai utilisé pour la première fois "GitHub Project", cela a été pratique car elle est très simple d'accès et suffisante à mes besoins.

Lien vers ma planification : <https://github.com/users/JulienETML-hub/projects/1/views/1>



Etat des lieux

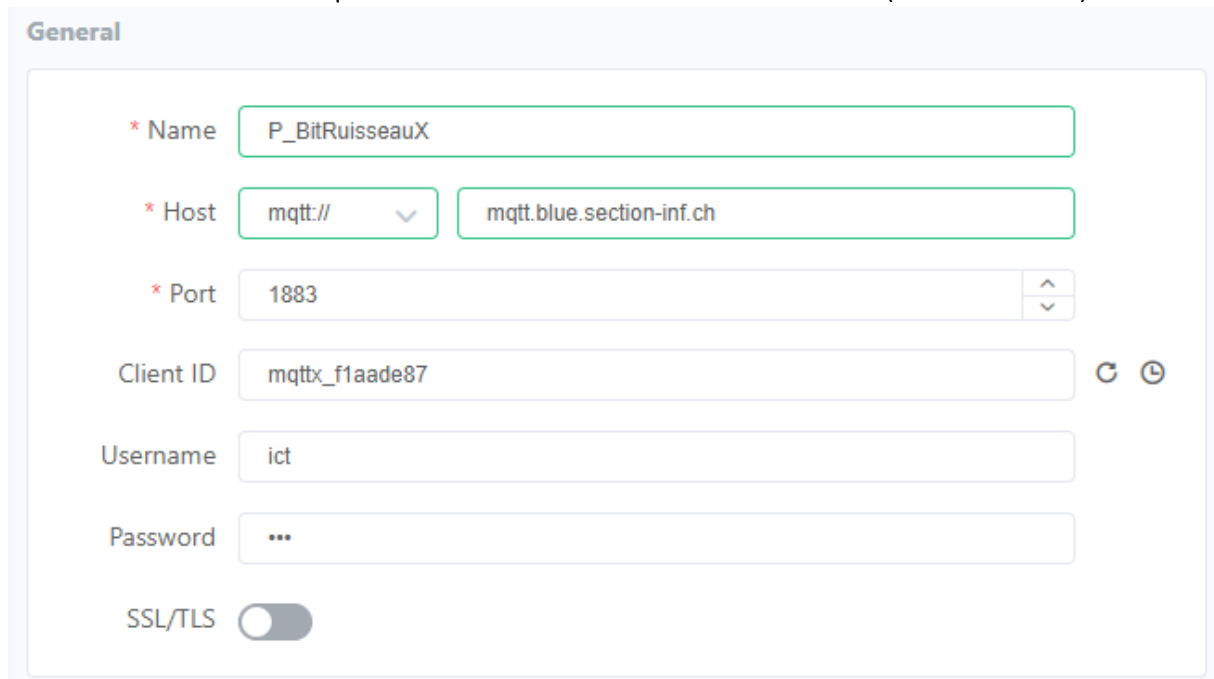
Actuellement mon programme permet la communication de catalogues, de la demande à l'envoi de catalogues ainsi que la demande et l'envoi de musique avec toutes ses informations ainsi que son contenu binaire. Une fois une musique reçue elle est directement ajouter dans un dossier sous forme de fichier .mp3 et prêt à l'écoute.

Mon programme permet aussi l'ajout d'une musique local au PC dans le catalogue de l'application. Cependant mon programme ne peut pas faire office de mediaplayer, il ne peut pas écouter une musique.

Démonstration d'utilisation (solo)

Configuration Mqttx

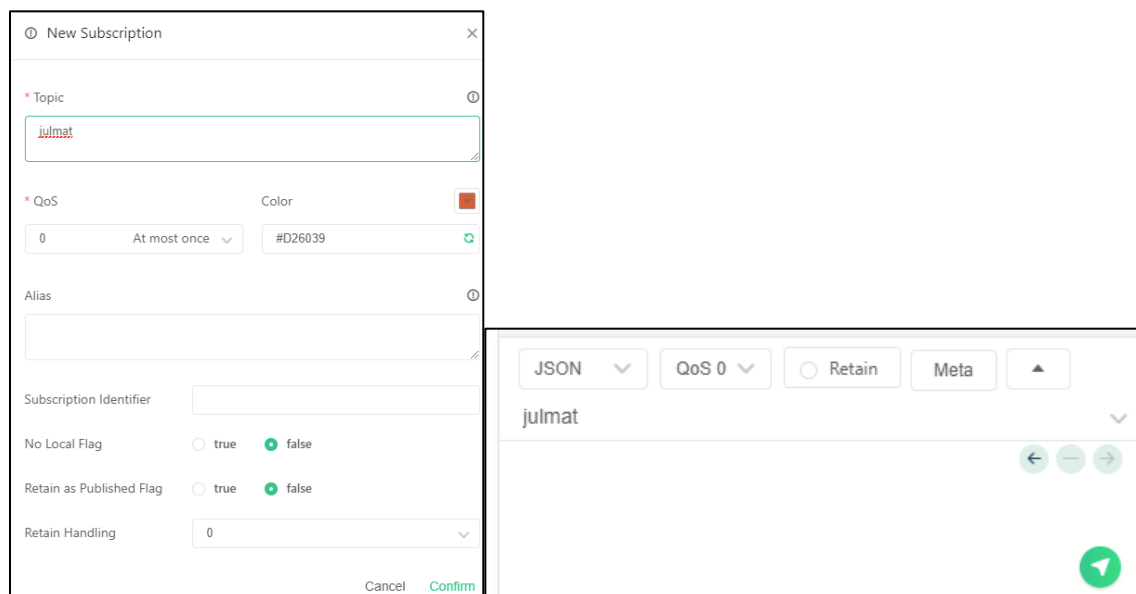
Premièrement il faut via mqttx se connecter au broker comme ci-dessous (Password = 321) :



The screenshot shows the 'General' configuration window for Mqttx. It contains the following fields and controls:

- Name:** P_BitRuisseauX
- Host:** mqtt:// (dropdown) and mqtt.blue.section-inf.ch (text input)
- Port:** 1883 (text input with up/down arrows)
- Client ID:** mqttx_f1aade87 (text input with refresh and save icons)
- Username:** ict (text input)
- Password:** ... (password input field)
- SSL/TLS:** A toggle switch currently turned off.

Puis s'abonner au topic "julmat", c'est celui qu'utilise mon programme pour l'instant.



The screenshot shows two overlapping windows from an MQTT client application.

The 'New Subscription' dialog on the left has the following settings:

- Topic:** julmat
- QoS:** 0 (At most once)
- Color:** #D26039
- Alias:** (empty)
- Subscription Identifier:** (empty)
- No Local Flag:** false (selected)
- Retain as Published Flag:** false (selected)
- Retain Handling:** 0

The MQTT client interface on the right shows:

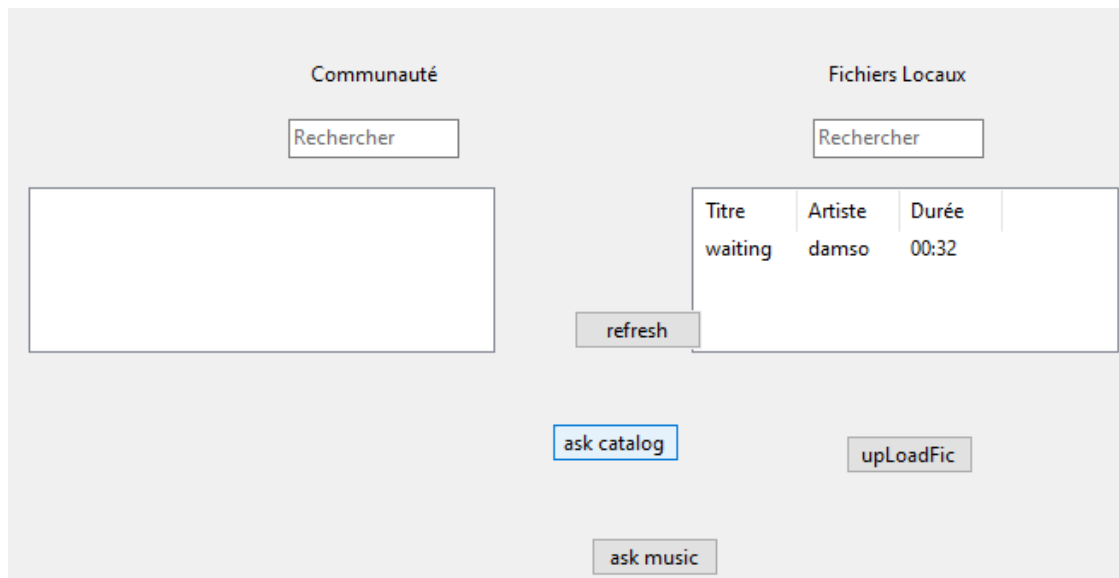
- Format:** JSON (dropdown)
- QoS:** 0 (dropdown)
- Retain:** (radio button)
- Meta:** (radio button)
- Topic:** julmat (text input with a dropdown arrow)
- Message History:** A list of messages with navigation arrows (back, forward, and a search icon).

Scénario de test du programme


Maintenant voilà plusieurs scénarios de test du programme qui permet de l'utiliser comme il se doit:

1. Demande de catalogue

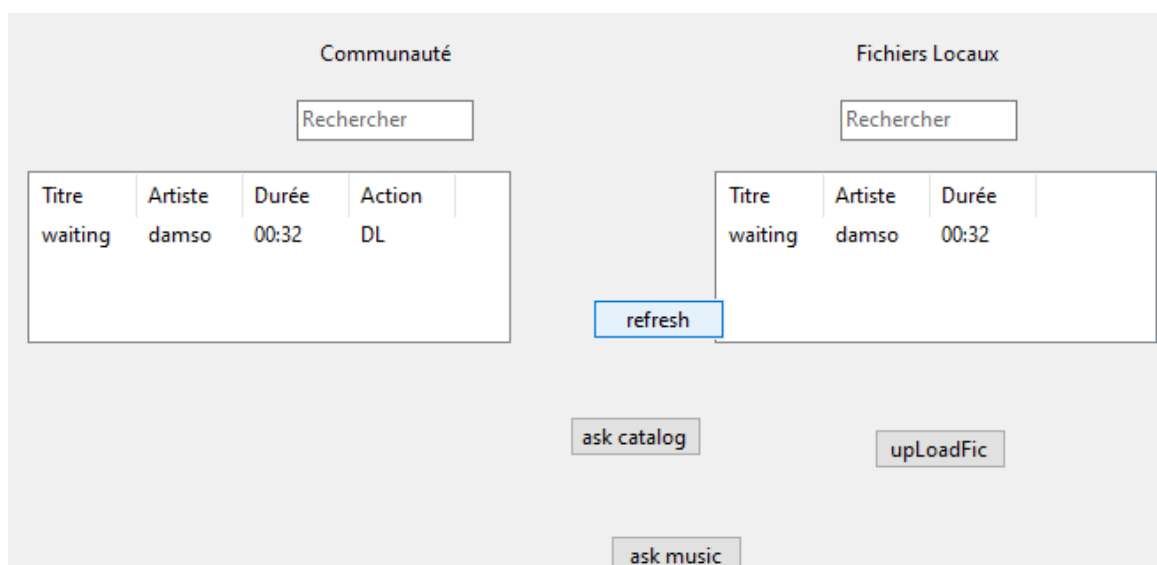
Une fois le programme lancé, cliquer sur "ask catalog", cela enverra sur le topic julmat un message de demande de catalogue, si vous souhaitez le tester seul, copier le message et modifier simplement un caractère du "SenderId" avant de le renvoyer :



2. Reception catalogue

Dans mon cas de test mon catalogue ne contient qu'une musique :  waiting.mp3

Une fois qu'on reçoit un message d'envoi de catalogue le programme le reçoit et le stock, il suffit d'appuyer sur refresh afin de mettre à jour la liste de catalogue reçu par la communauté :



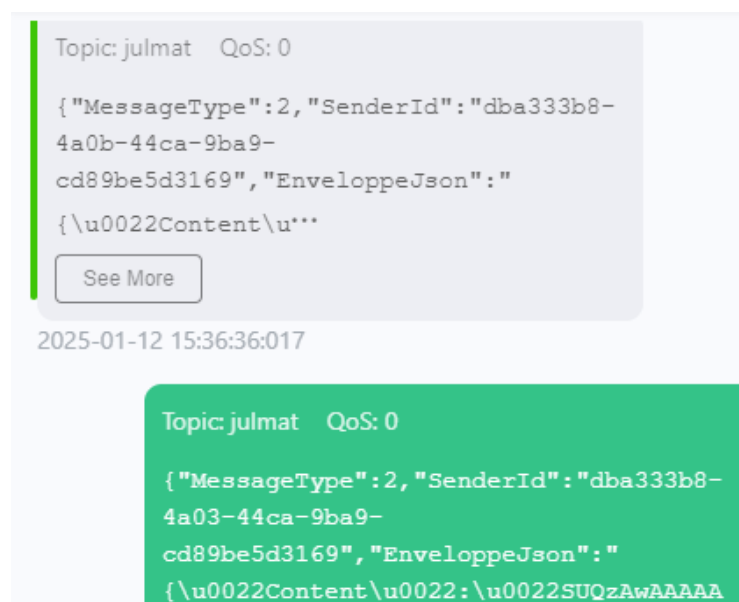
3. Demande de musique

Afin de demander une musique il suffit de cliquer sur "DL", cela va envoyer un message sur le topic. (Même chose qu'avant si vous testez seul il faut copier le message sur mqttx et le renvoyer avec un autre SenderId)




4. Réception musique

Une fois une demande de musique effectué vous allez recevoir les informations de la musique (Il faudra encore copier et modifier le SenderId du message) :



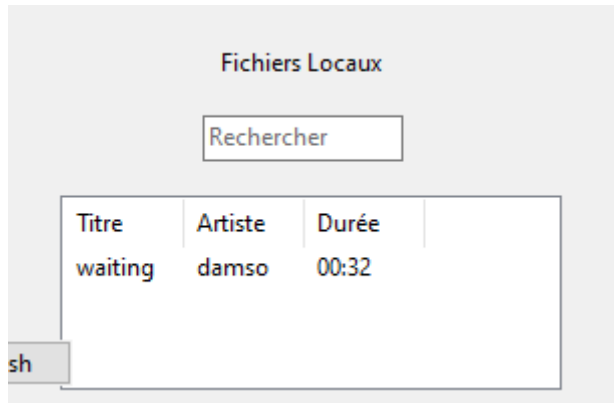
Une fois que c'est fait le fichier est correctement créé dans l'arborescence :

Documents > GitHub > JulMares-P_BitRuisseau > P_BitRuisseau > ressource > test				
Nom	N°	Titre	Interprètes ayant p...	Album
 waiting.mp3		waiting	damso	

A lire concernant les tests

Arborescence test solo

Si vous tester mon programme comme ci-dessus cela peut vous paraître étrange d'ajouter les fichiers des musiques dans un sous dossier "test" et pas directement dans ressource mais c'est simplement pour faciliter les tests en local (Sinon vu que j'importe des musiques que j'ai déjà on pourrait se demander si elles viennent vraiment du programme ou si elles étaient déjà là). Evidemment si je devais le mettre en production j'ajouterais les musiques directement dans "ressource" ce qui ferait en sorte qu'elles seraient aussi ajoutées dans la liste des musiques locales :



Modification SenderId

Dans mes tests il faut à chaque message envoyé aller modifier le SenderId sur mqttx mais évidemment que dans une vraie utilisation il n'y aurait pas besoin de ça puisque les communicants auraient un autre SenderId.

Clarifications des différents scénarios de test

Voici un tableau de scénario de test effectué ci-dessus plus concis (On part du principe que je communique avec une deuxième application compatible et donc je ne vais pas répéter à chaque fois que je modifie le SenderId sur mqttt).

Fonctionnalité	Etat	Action	Résultat
Demande catalogues	Programme lancé	Lorsque que je cliquer sur "ask catalog"	Cela envoie une demande broadcast et toutes les machines connectées m'envoient leurs catalogues
Réception catalogues	Programme lancé et reçoit divers catalogues (les musiques des catalogues vont automatiquement être ajouté à la liste "mediaDatasOnline"	Lorsqu'on clique sur refresh	Cela va mettre à jour la liste des musiques de la communauté qui est connecté à la liste "mediaDatasOnline", résultat on voit toutes les musiques reçues.
Demande fichier	On a bien reçu la liste des musiques et elle est visible dans la liste "Communauté"	Lorsque qu'on appuie sur "DL"	Cela va envoyer une demande de fichier concernant la musique correspondantes au bouton "DL"
Réception fichier	On reçoit une envoie de fichier d'une musique	Elle va automatiquement se créer dans l'arboresence	(Dans la version actuelle on peut seulement voir dans le dossier "test" que la musique a été DL mais si on modifie la route pour l'amener dans le dossier "ressource" on la verrait dans la liste "Fichiers locaux" sur l'UI (à condition que cette liste soit actualisée)

Utilisation AI dans ce projet

Dans ce projet j'ai utilisé chatgpt pour m'aider pour certaines questions (concernant la création de fichier et la conversion de string en byte un peu, mais cela s'est avéré plutôt simple. Concernant le gros du projet je n'ai pas beaucoup utilisé chatgpt car je ne pense pas qu'il aurait été d'une grande aide à moins d'avoir un excellent prompt du fait de la nature du projet, qui dépend beaucoup de notre environnement et de nos normes fixées en classe qui aurait été compliqué à expliquer à chatgpt. Pour résumé je l'ai utilisé pour des questions simples, il m'a fait gagner pas mal de temps mais n'a pas été décisif dans la création de cette application.

Conclusion

J'ai beaucoup aimé ce projet, comme pour la plupart des projets en programmation, j'ai dû un peu travailler à la maison mais cela ne m'a pas déplu, j'ai beaucoup apprécié le fait d'apprendre à communiquer des données via MQTT, la manière dont faire le cahier des charges est aussi très sympa je trouve et le fait de nous forcer à commencer par une analyse technique a été une bonne idée je trouve afin qu'on puisse mieux cerner les tenants et aboutissants du projet. En somme cela a été un très bon projet, avec une petite note négative quand même, j'aurais préféré que la question du protocole commun soit mieux gérée, par exemple en le définissant clairement au début et on y touchant plus jusqu'à la fin ou en faisant des groupes devant communiquer entre eux. Car dans ce projet j'avais l'impression qu'on devait suivre des personnes qui eux pouvaient modifier leurs codes chaque semaine, ce qui je trouve est plutôt éprouvant et inégal.

Journal de travail

Journal de travail : [P_BitRuisseau_jdt_JulienMares.xlsx](#)