AudioWave

Maintenance Guide

Submitted in partial fulfillment for the degree of

Bachelor of Science

in

Software Engineering

for

The Software Engineering Department

at

Braude - College of Engineering, Karmiel

By

Ward Zidani

Ahmad Bsese

Under guidance of

Zeev Frenkel

Project code

24-1-D-44

Summer 2024

Table of Contents

1. Introduction

- Project Overview: Audiowave is a full stack project for sharing and consuming audios for free. It is built in a microservice style architecture, with multiple services with different responsibilities in the backend, and a mobile client application in the frontend.
- Technologies:
    - General:
        - Git (Bash, Github)
        - Postman
    - Client-side:
        - Flutter (Dart)
        - FFmpeg
    - Server-side:
        - .Net Core 8 (C#)
        - RabbitMq
        - Nginx API Gateway
        - AWS S3
        - Atlas (MongoDb)
        - Microsoft SQL Server 2022
        - Docker
        - Docker-Compose

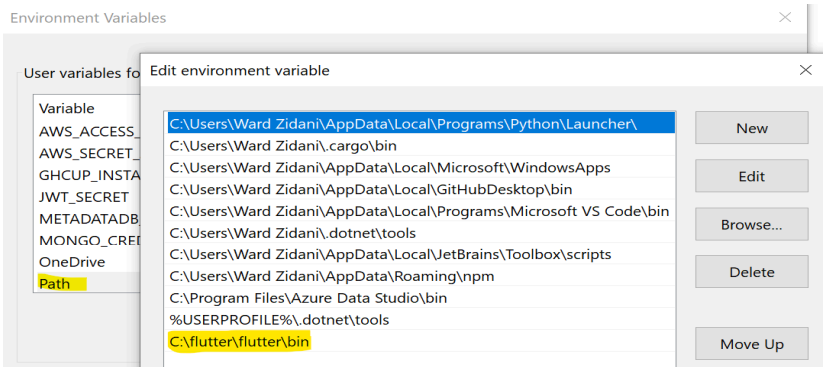
2. Getting Started

- Prerequisites:
    - Docker Engine (Docker Desktop)
    - Visual Studio 2022
    - Visual Studio Code
    - Microsoft SQL Server
    - Microsoft SQL Server Management Studio 2022
    - Android Studio
    - Flutter (Dart)

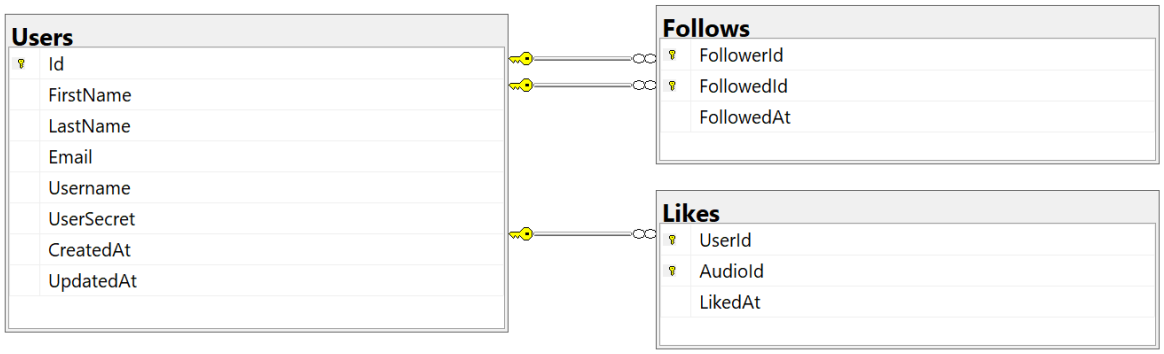
3. Setup

1. Install all programs from the “Prerequisites” section.
2. Flutter:

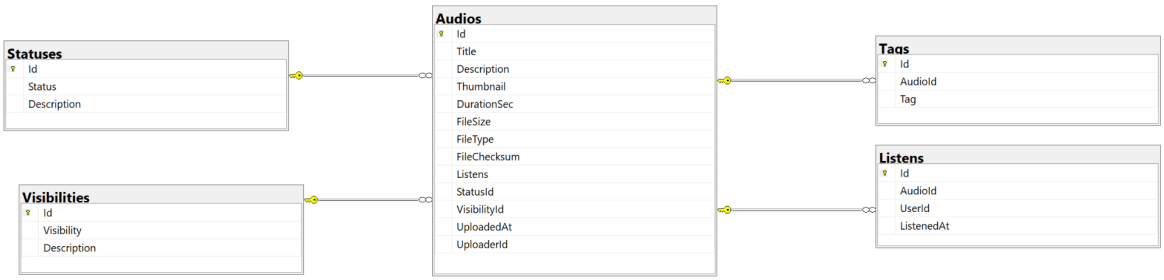
i. When downloading the Flutter zip file, create a folder named “flutter” in your C:/ directory so it looks like this: “C:/flutter/”
ii. Extract the downloaded flutter folder into this directory so it looks like “C:/flutter/flutter/*”. * is just the contents.
iii. Open Environment variables in your machine.
iv. Navigate to “Path” (Create if not exist)
v. Add a row “C:\flutter\flutter\bin”



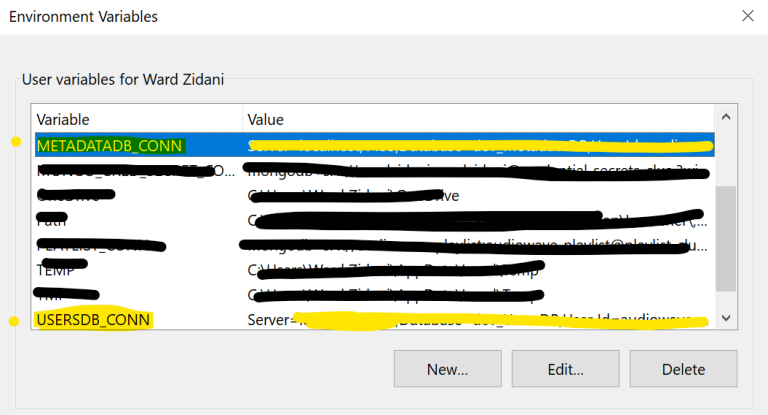
3. Microsoft SQL Server:
i. Create a database and name it “dev_UsersDB” or something similar.



ii. Add the tables in the “Users Create.sql” file in the submission folder.
iii. Create a database and name it “dev_MetadataDB” or similar.
iv. Add the tables in the “Metadata Create.sql” file in the submission folder.

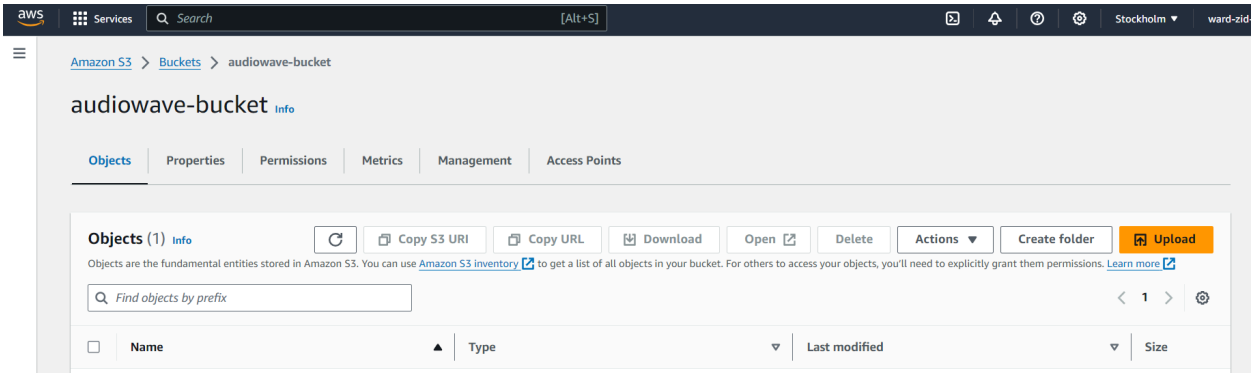


v. Once both Databases are set up, fetch their connection strings and add them into your environment variables

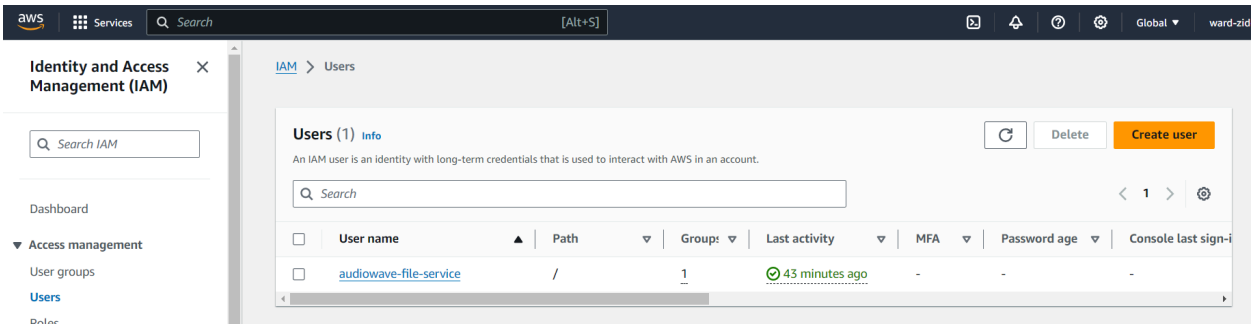


4. AWS S3:

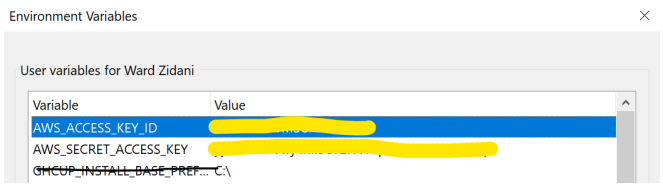
i. Create an AWS S3 Bucket and name it “audiowave-bucket”



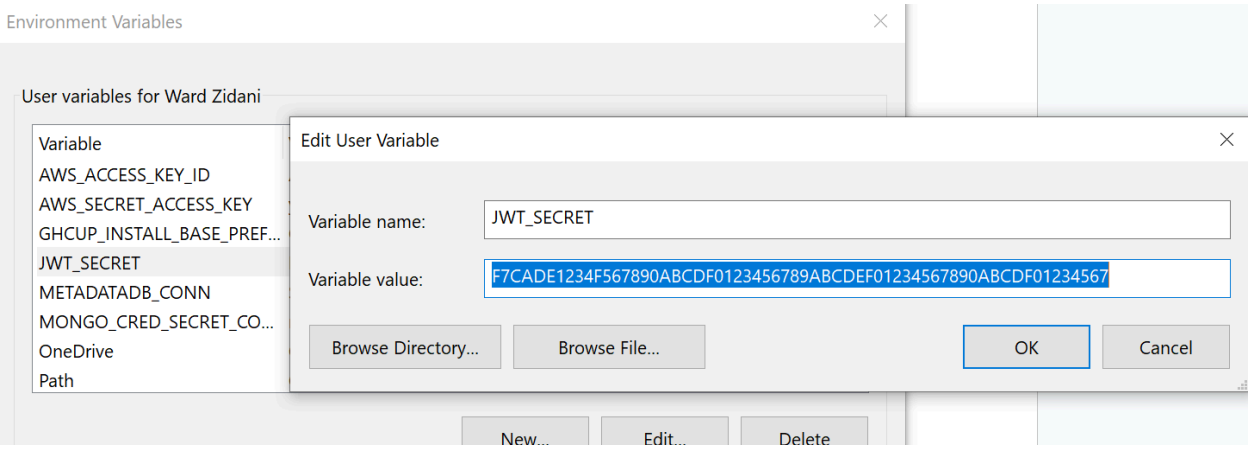
ii. Create an AWS IAM user for the project and give it permission for the previously created S3 bucket



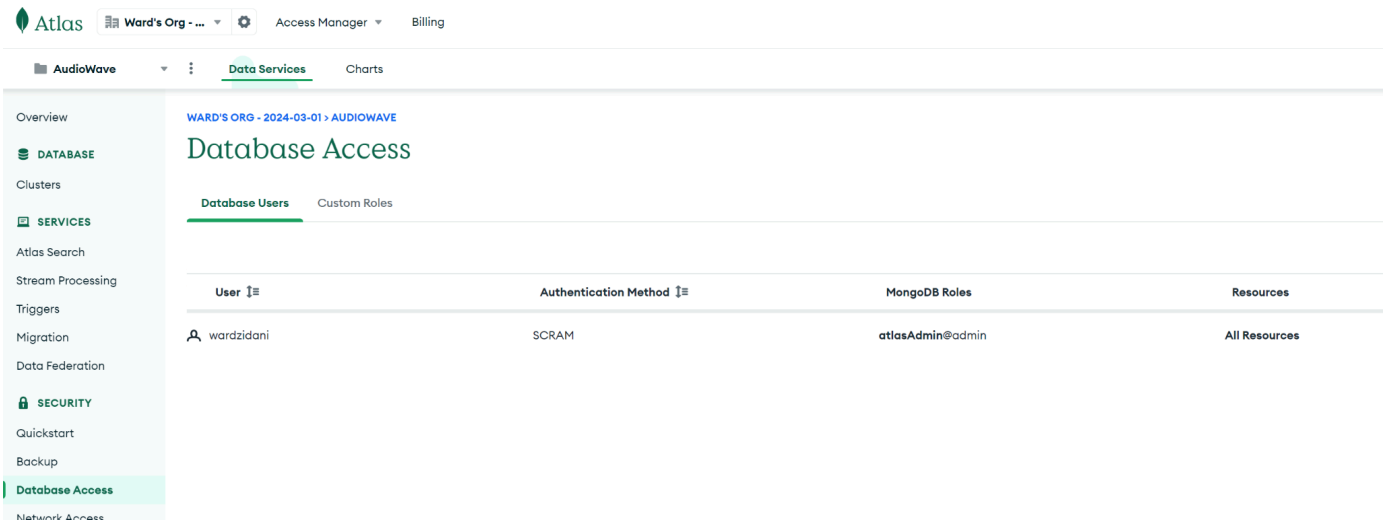
iii. Fetch the AWS Access Key and AWS Secret Access key and add them to the environment variables



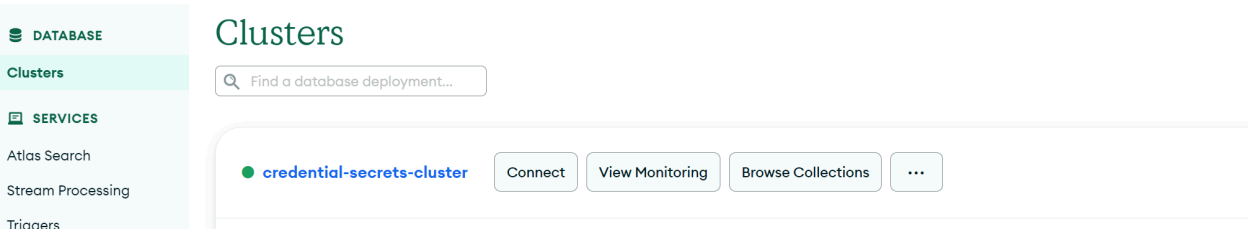
5. Add a JWT Secret to your environment variables
    i. Value : **F7CADE1234F567890ABCDF0123456789ABCDEF01234567890ABCDF01234567**



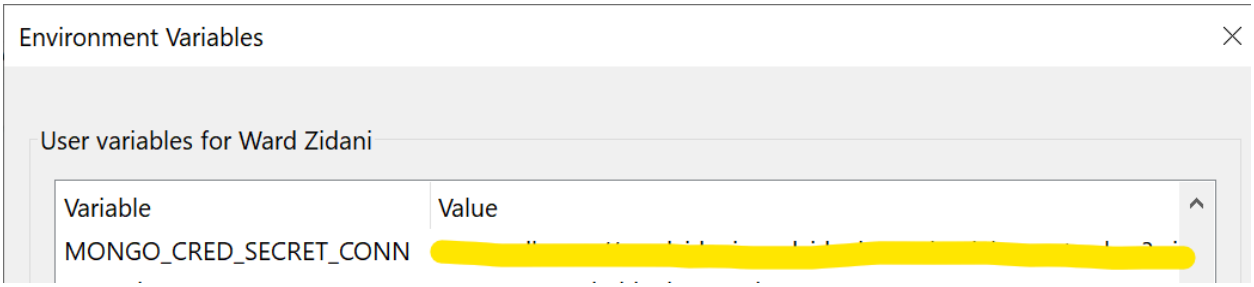
6. Credential Encryption Key Storage
    i. Open Atlas MongoDB and Create a project
    ii. Create a free M0 free cluster
    iii. Create a user for that Cluster



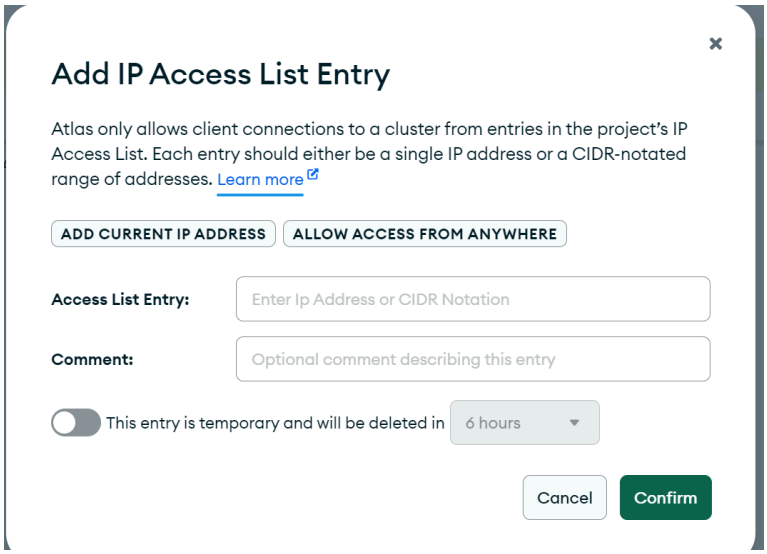
    iv. Get the connection string from the atlas by clicking “Connect” on the cluster



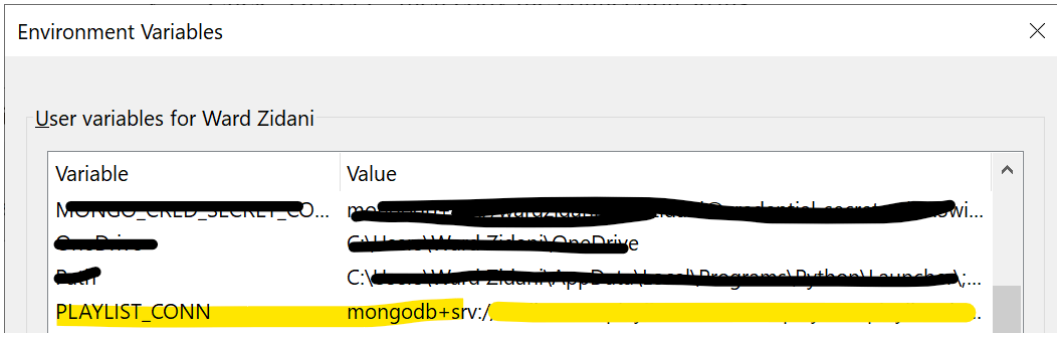
v. Click “Drivers”, then copy the connection string.
vi. Paste the connection string into the environment variables.



vii. In Atlas, Navigate to network access, add ip and choose “Allow Access From Anywhere” and confirm.



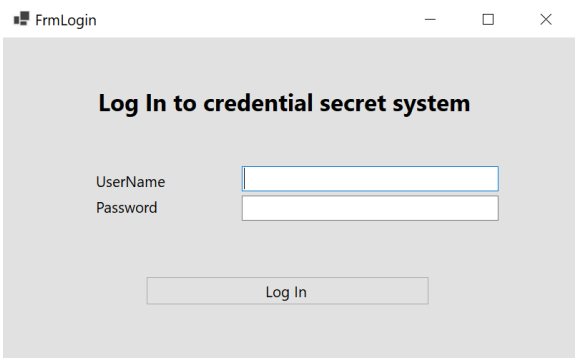
7. Playlist Database
   i. Follow the same exact steps from section 6 “Credential Encryption Key Storage”, but name is you see fit.
   ii. As before, paste the connection string into the environment variables.



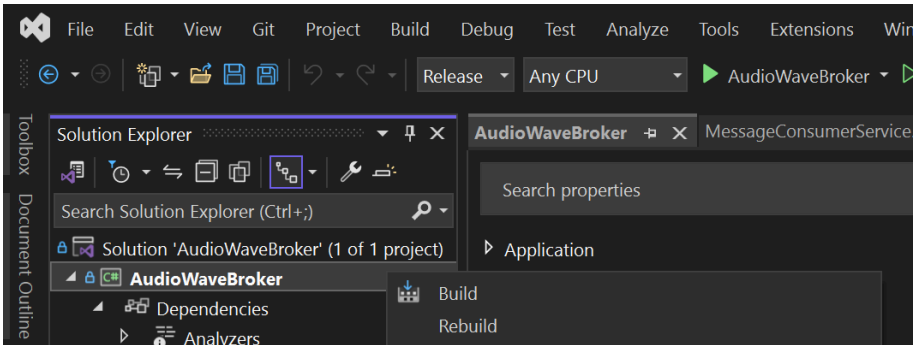
8. Credential Encryption Key Generator
    i. Using Visual Studio 2022, open the CredentialSecretsApp from the Desktop directory the the submission root.
    ii. In the login form, add the MONGO_CRED_SECRET_CONN connection string that we added earlier as such

```
var loginForm = new FrmLogin();
while (isConnected == false)
{
    var dialogResult = loginForm.ShowDialog();

    if (dialogResult == DialogResult.OK)
    {
        connectionUri = $"mongodb+srv://{loginForm.Username}:{loginForm.Password}@credential-secrets-clus
        InitializeMongoClient();
        CheckConnection();
    }
    else
    {
        break;
```

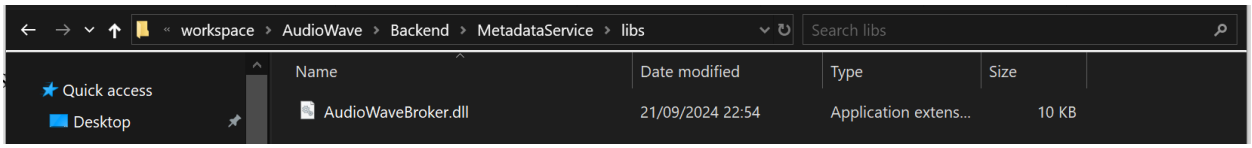
    iii. Run the program and enter your Atlas user credentials for the Encryption Credentials project
    iv. Generate New keys as desired
    v. The keys will automatically be stored in the Secret Key store, the preview is just aesthetic



9. RabbitMq
    i. Open Visual Studio 2022
    ii. From the root directory of the repository, open the solution in "AudioWave\Backend\libraries\AudioWaveBroker\AudioWaveBroker.sln"
    iii. Enure all the needed Nuget packages are installed, if not, install them
    iv. Right Click the Solution or project root directory and choose "Build"

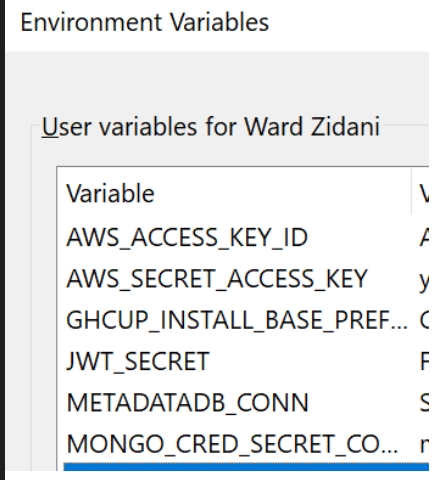


v. Now, in each one of the backend services, there should be a file named libs and inside it an audiowavebroker dll.



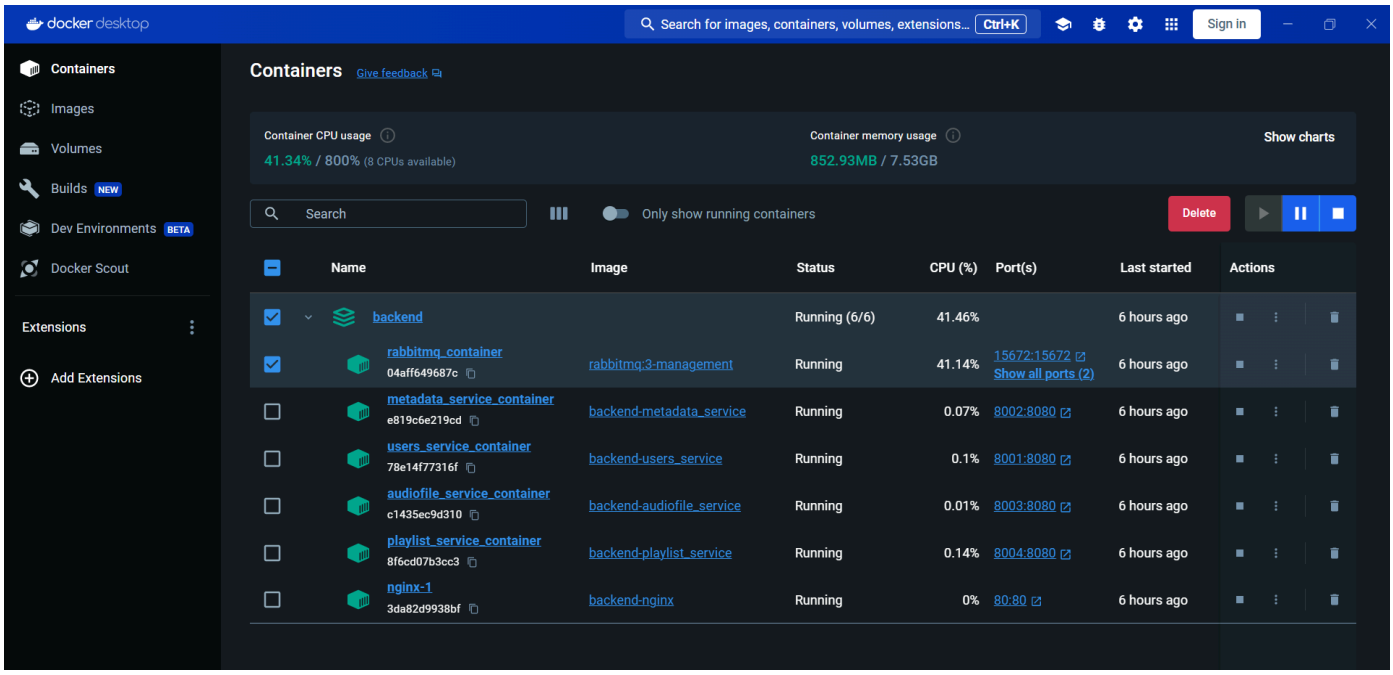
10. Running Backend Services
    i. For each backend service (AudioFileService, MetadataService, PlaylistService, and UsersService), open each solution, one-by-one and check that all the Nuget packages are installed as needed.
    ii. Ensure the Docker Engine has started on your machine (Docker Desktop is a helpful tool to do that for you).
    iii. Go to the Backend directory in the repo and open the "docker-compose.yml" file in a text editor
    iv. Replace each placeholder with the appropriate value from the environment variables. Each environment variable's name should directly correspond to the keys in the yaml file.

```yaml
metadata_service:
  build: ./MetadataService
  container_name: metadata_service_container
  environment:
    - METADATADB_CONN=<your metadata db connection string>
    - JWT_SECRET=<your jwt secret>
  ports:
    - "8002:8080"
  depends_on:
    - rabbitmq
  networks:
    - backend-network

audiofile_service:
  build: ./AudioFileService
  container_name: audiofile_service_container
  environment:
    - AWS_ACCESS_KEY_ID=< your aws access key >
    - AWS_SECRET_ACCESS_KEY=<your aws secret key>
    - JWT_SECRET=<your jwt secret>
  ports:
    - "8003:8080"
  depends_on:
    - rabbitmq
  networks:
    - backend-network

playlist_service:
```

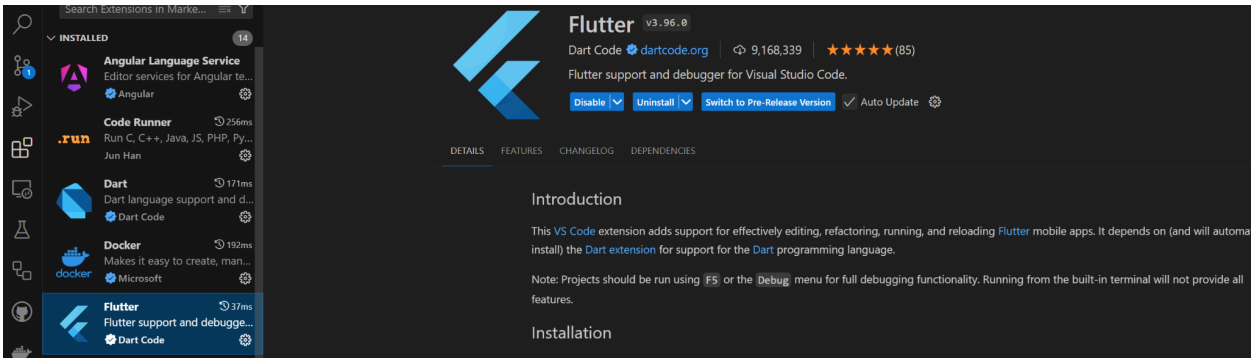


v. Finally, open Command Prompt and navigate to the Backend directory on your machine.
vi. Type the command "docker-compose up —build" and hit Enter.

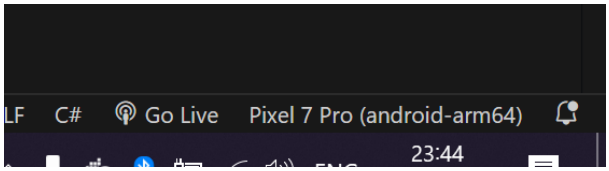
vii. If all was successful, the Docker Desktop program will show 6 running containers



11. Running Frontend Debugging
    i. Open Visual Studio Code
    ii. Download the Flutter Extension

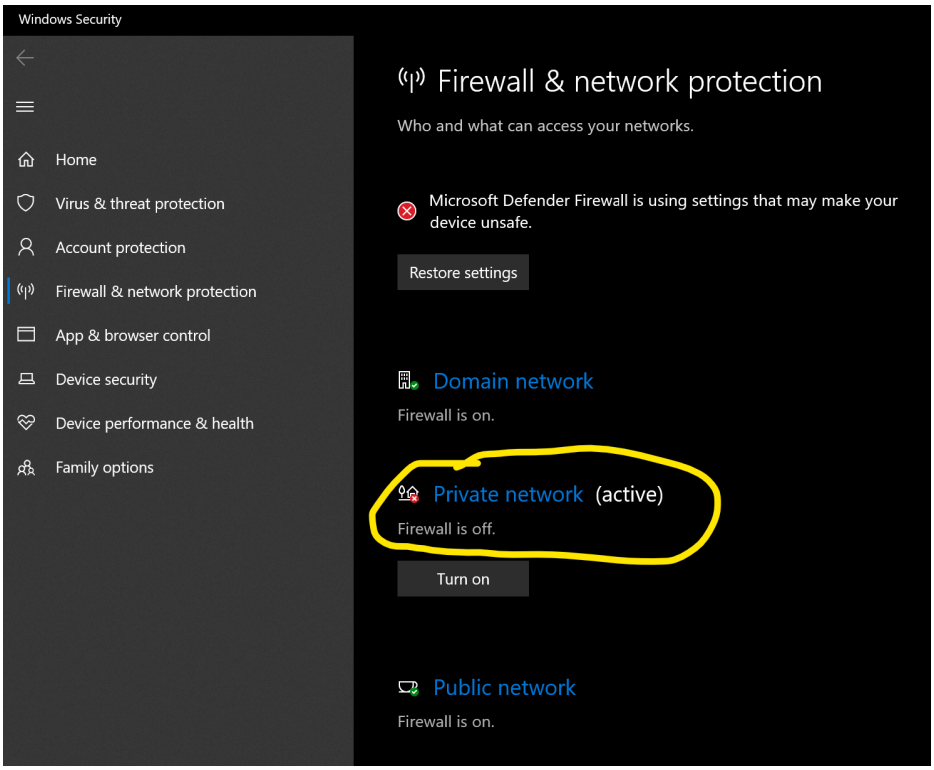


iii. Open the project in the repo at “AudioWave\Frontend\audiowave6”
iv. In the terminal (in the same directory) type “flutter pub get” and run, this will install all the needed dependencies to your machine.
v. On an android development mobile device, turn on developer mode and USB debugging (for instructions CLICK HERE) and plug your device into the PC
vi. The device should show up in the visual studio code window at the bottom right.



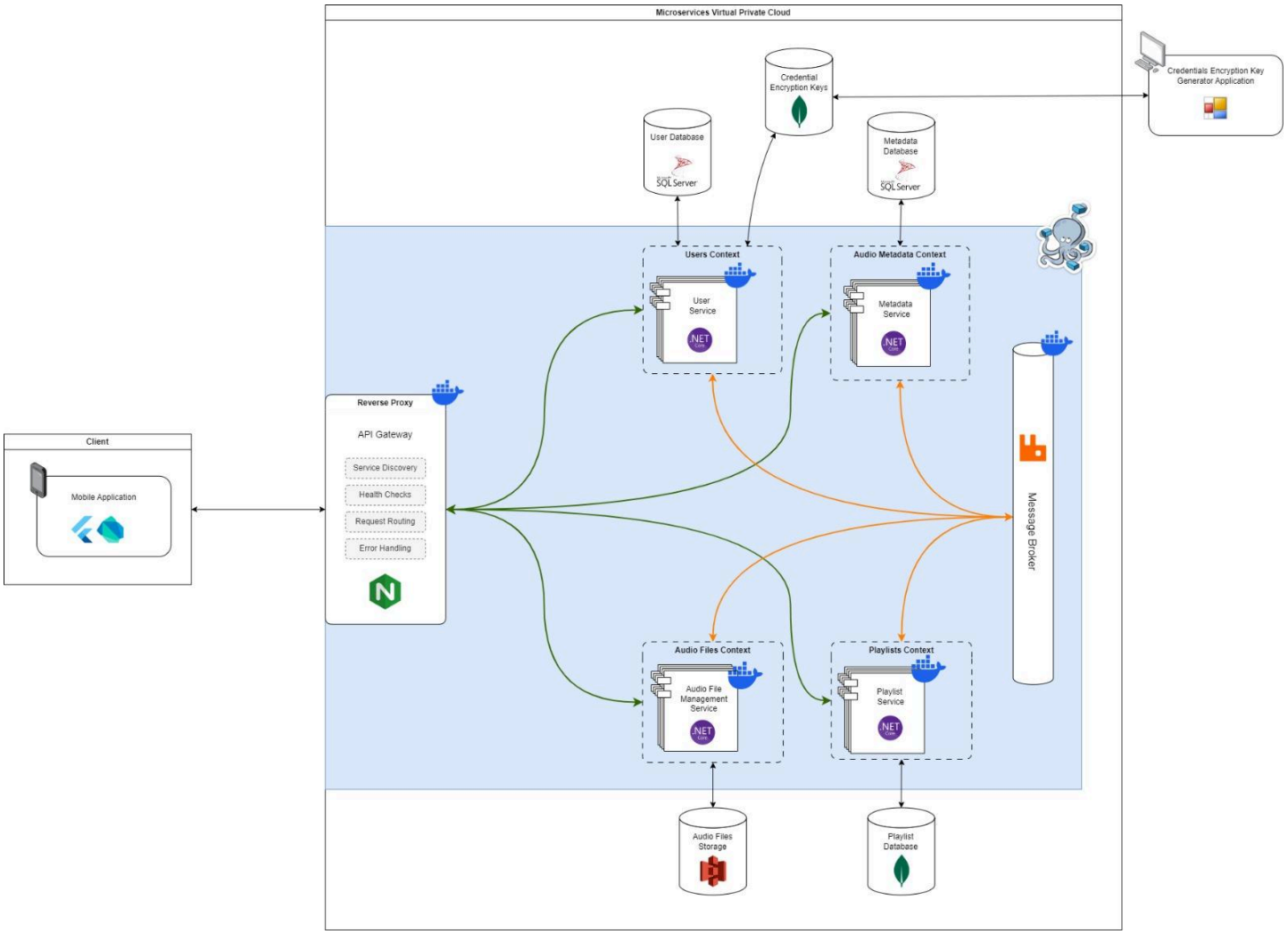
vii. Go to the file “AudioWave\Frontend\audiowave6\lib\data\api\endpoints.dart” and set the ip to your personal computer’s private network IPV4 address (for instructions to find your ip CLICK HERE).

viii. In your PC, make sure to disable the firewall in your private network.



ix. Back in visual studio code, in the terminal, type “flutter run” and hit Enter.

4. Architecture



The Architecture in this project is all based around the idea of splitting the backend into smaller, more manageable services, as can be seen in the diagram.