Code source
- Flutter
- Jukebox API
- MusicGen VM Service
- Agent AI

How to?
- Flutter
- Jukebox API
  - Firestore
  - Run & Deploy
- MusicGen VM Service
- Agent Ai

Google Jukebox

Code source

Flutter

Jukebox API

https://storage.googleapis.com/bigdata_code/fr-google-jukebox.zip

MusicGen VM Service

https://storage.googleapis.com/bigdata_code/fr-google-jukebox-musicgen.zip

Agent AI

How to?

Flutter

The **JukeBox** application was developed using Flutter to deliver a seamless and high-performance user experience. The user interface (UI) was specifically designed and optimized for the **Google Pixel Tablet 9**, leveraging its large screen for a clean and ergonomic design. The application is available as an APK, securely hosted in a bucket within the **Google Cloud Platform (GCP)** project, ensuring easy deployment and access.

The state management in the application is implemented using **BLoC (Business Logic Component)**, ensuring a robust and scalable architecture. Additionally, **isolates** are employed for efficient communication with the backend during the music generation process, allowing for smooth performance without blocking the main thread. The application is available as an APK, securely hosted in a bucket within the **Google Cloud Platform (GCP)** project, ensuring easy deployment and access.

Download apk here : https://storage.googleapis.com/bigdata_apk/jukebox.apk
Download Source code here: https://storage.googleapis.com/bigdata_code/jukebox.zip

Jukebox API

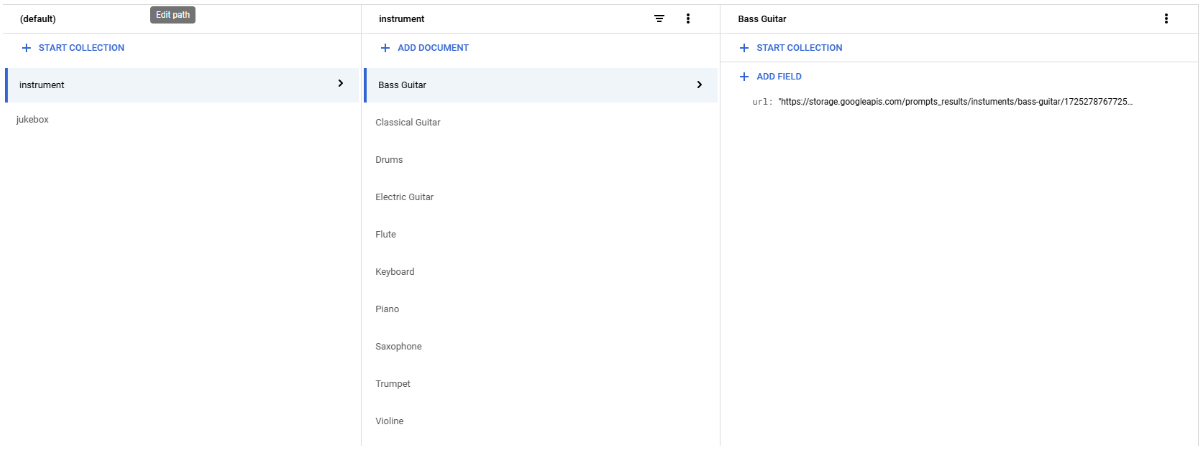
https://storage.googleapis.com/bigdata_code/fr-google-jukebox.zip

This is the api endpoint that the Flutter application is calling for all its functionality.

Firestore

Prepare two collections in the default db.

1. instrument : Collection of possible instruments with
   a. KEY= Instrument name
   b. VALUE = {url: image url of the instrument}



(default): instrument, jukebox

instrument: Bass Guitar, Classical Guitar, Drums, Electric Guitar, Flute, Keyboard, Piano, Saxophone, Trumpet, Violine

Bass Guitar:

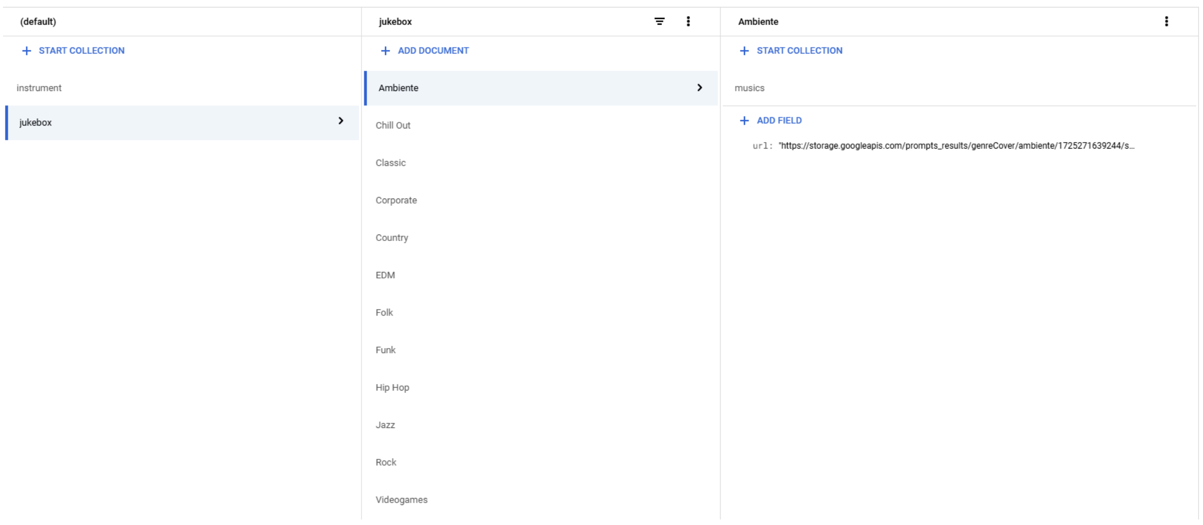
url: "https://storage.googleapis.com/prompts_results/instuments/bass-guitar/1725278767725...

Here are the values (as they will be presented by the api, not as in firestore)

None

```
 {
      "id": "Bass Guitar",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/bass-guitar/1725278767725/sample_0.png"
    },
    {
      "id": "Classical Guitar",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/classical-guitar/1725278673249/sample_0.png"
    },
    {
      "id": "Drums",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/drums/drums.png"
    },
    {
      "id": "Electric Guitar",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/electric-guitar/electric-guitar.png"
    },
    {
      "id": "Flute",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/flute/1725277638668/sample_0.png"
    },
    {
      "id": "Keyboard",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/keyboard/1725278828795/sample_0.png"
    },
    {
      "id": "Piano",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/piano/piano.png"
    },
    {
      "id": "Saxophone",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/saxophone/saxophone.png"
    },
    {
      "id": "Trumpet",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/trumpet/trumpet.png"
    },
    {
      "id": "Violine",
      "url":
"https://storage.googleapis.com/prompts_results/inst
uments/violine/violine.png"
    }
```

2. jukebox: Collection of possible instruments with
    a. KEY: Music genre
    b. VALUE: { url: imageURL of genre, musics: subcollection with the created songs via the application}



(default)
instrument
jukebox

jukebox
Ambiente
Chill Out
Classic
Corporate
Country
EDM
Folk
Funk
Hip Hop
Jazz
Rock
Videogames

Ambiente
musics
url: "https://storage.googleapis.com/prompts_results/genreCover/ambiente/1725271639244/s...

Here are the values (as they will be presented by the api, not as in firestore)

```None
{
      "id": "Ambiente",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/amb
iente/1725271639244/sample_0.png"
    },
    {
      "id": "Chill Out",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/chi
ll-out/1725270439934/sample_0.png"
    },
    {
      "id": "Classic",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/cla
ssic/1725269390232/sample_0.png"
    },
    {
      "id": "Corporate",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/cor
porate/1725271180300/sample_0.png"
    },
    {
      "id": "Country",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/cou
ntry/1725269867234/sample_0.png"
    },
    {
      "id": "EDM",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/dan
ce-edm/1725270098480/sample_0.png"
    },
    {
      "id": "Folk",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/fol
k/1725269597317/sample_0.png"
    },
    {
      "id": "Funk",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/fun
k/1725270593601/sample_0.png"
    },
    {
      "id": "Hip Hop",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/hip
-hip/1725271363924/sample_0.png"
    },
    {
      "id": "Jazz",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/jaz
z/1725269657228/sample_0.png"
    },
    {
      "id": "Rock",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/roc
k/1725269730725/sample_0.png"
    },
    {
      "id": "Videogames",
      "url":
"https://storage.googleapis.com/prompts_results/genreCover/vid
eogames/1725270824304/sample_0.png"
    }
```

Run & Deploy

Prequisite:

- Gemini API key: https://aistudio.google.com/app/apikey
- MusicGen VM Service endpoint (see below instructions to create the VM)
- *Optional*: Google Account + App Password (https://knowledge.workspace.google.com/kb/how-to-create-app-passwords-000009237) to send emails to users

1) **Configure secrets**
   Configure two secrets
   - GEMINI_API_KEY → Gemini key
   - GOOGLE_APP_PASSWORD —> App password for the account to send emails (*Optional*)
2) **Run locally**
   Follow the instructions in the README. Make sure that
   - .env is present (see README)
   - Firestore is setup
   - Secrets for set
   - If you want to create Music the VM must be running and accept connections
3) **Deploy the service via Cloud run**
   a) From the root of the code run

```None
gcloud run deploy jukebox --source . --set-env-vars
"MUSICGEN_URL=http://<vm-name>.<zone>.c.<project_id>.internal:
8000" --set-env-vars "GOOGLE_APP_EMAIL=<email_address>"
--service-account=<sercive_account_to_run_the_cloud_run>
--min-instances=1  --network=default  --subnet=default
--vpc-egress=private-ranges-only --region=europe-west1
```

Note:Make sure you add the right external URL for the MusicGen VM.
You have to adjust the MusicGen VM name and zone depending on your Google Cloud configuration (used in below configuration for the MusicGen backend)
For example:
`“MUSICGEN_URL=http://jukebox-instance.us-central1-f.c.dgc-ai-jukebox.internal:8000”`

MusicGen VM Service

This is the VM that will create the Songs via the MusicGen model.

https://storage.googleapis.com/bigdata_code/fr-google-jukebox-musicgen.zip

1) **Create Firewall rule**
   Create a http firewall rule that allows access of the backend
   - Name: http-server
   - Target-tags: http-server
   - Source ipv4 ranges: 0.0.0.0/0
   - Protocol and Ports
     - TCP
     - 8000

2) **Create VM Instance**

   Create a VM instance with the following specs:

   - name : *jukebox-instance*
   - n1-standard-4
   - 2x Nvidia T4 GPUs
   - Boot Disk
     - SSD
     - 100GB
     - Image: *Debian 11, Python 3.10. With CUDA 11.8 preinstalled* (or similar)
   - Networking:
     - Allow http-traffic
     - Network-tags: http-server
   - Service Account Cloud API access scopes:
     - Allow read/write on storage api

   Note:
   Name the instance to *jukebox-instance*. This is important for the other service to to correctly identify the instance's private hostname.If you choose a different name, make sure to update the frontend Cloud Run env variable with the appropriate private hostname.

3) **Environment setup**
   Once your VM is up and running, you'll need to set up the necessary environment and dependencies.

   a) Install GPU Dependencies
      When you first connect to the VM via ssh, you will be prompted to install the necessary GPU-related dependencies.

      - Type “y” when prompted.

   b) Run the Setup Script
      Run the following in a terminal to set up the environment:

```
None
            set -e

            sudo mkdir /home/jukebox
            sudo chmod 777 -R /home/jukebox

            export HOME=/home/jukebox
            cd ~

            wget
            https://repo.anaconda.com/miniconda/Miniconda3-latest-Lin
            ux-x86_64.sh -O Miniconda3-latest-Linux-x86_64.sh
            bash Miniconda3-latest-Linux-x86_64.sh -b
            rm Miniconda3-latest-Linux-x86_64.sh
            source ~/miniconda3/etc/profile.d/conda.sh

            conda create --name jukebox_env python=3.9 -y
            conda activate jukebox_env

            conda install pytorch==2.1.0 torchvision torchaudio
            cudatoolkit=11.8 "ffmpeg<5" -c conda-forge -c pytorch -c
            nvidia -y
```

This script will:

- Create a directory for the Jukebox service that can be accessed by other users.
- Install Miniconda for managing Python environments.
- Create and activate a Conda environment named jukebox_env.
- Install PyTorch with GPU support

4) Unzip code and install dependencies

Upload fr-google-jukebox-musicgen.zip to the VM
Unzip the code into */home/jukebox/fr-google-jukebox-musicgen*

```
None
      unzip fr-google-jukebox-musicgen.zip -d /home/jukebox
```

Make sure the conda env is activated

```
None
export HOME=/home/jukebox
cd ~
conda activate miniconda3/envs/jukebox_env
```

Install dependencies

```
None
cd fr-google-jukebox-musicgen
pip install -r requirements.txt
```

5) **Setup google storage and select model**

   a) Setup a google storage bucket to save the audio files
      i) Add the service account that runs your VM to the bucket users
   b) Add the bucket name to the *.env* file
   c) Select a different model if needed for example: [facebook/musicgen-small]()
      i) Only the name is needed → see example in code or huggingface

```
None
MODEL_NAME=facebook/musicgen-large
BUCKET_NAME=<your-bucket-name>
```

6) **Run service**

Start the server

```
None
uvicorn main:app --host 0.0.0.0 --port 8000
```

This will take a while to download the model and load it into GPU memory

Find the docs here:
<public_ip_of_vm>:8000/docs

Note:
If there are problems finding a cache make sure the application has the necessary permissions to write to the current directory

```
None
      sudo chmod 777 -R /home/jukebox
```

Startup script (optional)

Add the following under *Automation* -> *Startup* script in the VM settings:

```
None
#!/bin/bash

# Export HOME environment variable
export HOME=/home/jukebox

# Change directory to home
   cd ~

# Change permissions of the /home/jukebox directory
sudo chmod 777 -R /home/jukebox

# Activate the Conda environment
source ~/miniconda3/bin/activate jukebox_env

# Go into the code folder
cd fr-google-jukebox-musicgen

# Start the uvicorn server and redirect stdout and stderr to a
log file
uvicorn main:app --host 0.0.0.0 --port 8000 >
jukebox_service.log 2>&1 &
```

This script will:

- Set the home Path
- Allow write access
- Activate conda env
- Start the fastapi service and write the logs into a seperate file jukebox_service.log

Agent Ai