

# Documentation du projet Jambox

N'attendez plus pour être créatif



## Sommaire

<b>Présentation du projet</b>	<b>1</b>
<b>Présentation de l'équipe et des méthodes de travail</b>	<b>2</b>
<b>Business Model &amp; Plan</b>	<b>4</b>
<b>Stratégie de communication</b>	<b>5</b>
<b>La Jambox (IOT)</b>	<b>9</b>
<b>IA - Lecture et génération de musique</b>	<b>10</b>
<b>API REST</b>	<b>14</b>
<b>Application mobile</b>	<b>15</b>
<b>Déploiement</b>	<b>19</b>
<b>Pour conclure</b>	<b>20</b>
<b>Sources</b>	<b>21</b>

## Présentation du projet

La Jambox est le compagnon idéal pour tout musicien rêvant de jouer en groupe durant ces temps d'isolation. Il suffit de la brancher et d'appuyer sur le bouton 'Allumage' afin de la lancer.

La box est équipée d'une entrée Jack femelle et d'une sortie Jack femelle afin de pouvoir brancher votre instrument et d'écouter vos productions sur une sortie son.

Vous pourrez également appairer votre Jambox à votre smartphone par Bluetooth, et choisir le mot de passe dès votre première connexion à partir de notre application mobile en vous munissant du code matériel de la Jambox (cf ci-dessous).

Afin d'utiliser l'application, il vous faudra créer un compte Jambox et entrer vos informations dans l'onglet 'créer un compte', vous pourrez également utiliser une inscription avec votre compte Google si vous en avez déjà un.

Vous pourrez rechercher par bluetooth votre Jambox et, avec le code présent sur votre emballage, valider l'appairage entre votre machine et votre smartphone

Une fois que vous êtes connectés à l'application Jambox et que vous êtes appairés, vous pouvez entrer vos paramètres de session :

- Vos différents instruments (basse uniquement pour le moment)
- Votre style de musique
- Votre rythme de session ou un rythme prédefini
- Activer/Désactiver l'enregistrement
- Écouter/Partager vos sessions

You êtes maintenant prêt à Jamer !

## Présentation de l'équipe et des méthodes de travail

L'équipe Jambox originellement composée de six membres ne présente plus que 5 collaborateurs aujourd'hui :

- Nathan Ragonnet, project manager et développeur Api - User
- Matthieu Daugas, développeur IA
- Bastien Berger, développeur IA & Coud
- Nelson Angelique, développeur mobile
- Louis Vince, développeur

Pour les méthodes de développement nous avons choisi de fonctionner avec la Méthodologie AGIL avec quelque modification qui vienne de la Méthodologie SCRUM.

En effet, toutes les 1 à 3 semaines (selon la taille des sprints) nous organisons une réunion de préparation de sprint. Toutes les personnes travaillant sur le sprint se doivent de participer aux réunions de préparation de sprint.

Pendant celle-ci beaucoup de choix importants sont faits:

- Les requirements qui devront être développés (en accord avec le planning de développement)
- Le nombre d'heures de travail correspondant au requirement qu'il faut développer
- Les ressources nécessaires pour la réalisation du sprint
- Les aides qui peuvent être mises en place pour la réalisation du sprint.

Une fois cette réunion faite, un dispositif sur l'application de management de projet Monday est mis en place. On peut y retrouver les tâches faites pendant le sprint et qui y est assigné. Toute les semaines est organisée une réunion portant sur l'avancée du sprint en cours. Pendant cette réunion sont abordés plusieurs points importants comme:

- L'avancée sur les tâches réparties
- Le retard ou non pris sur le sprint
- Compte rendu des ressources mises en place sur le sprint

Enfin à la fin du sprint une réunion pour parler du sprint est organisée (souvent suivie par une réunion de début de sprint). Pendant cet événement nous parlons à tour de rôle pour collecter les avis positifs et négatifs sur le sprint écoulé. Des notes sont prises pour faire évoluer de façon constructive le déroulé des prochains sprints. Un bref résumé du sprint est fait sur l'avancement et le retard de celui-ci.

Pour le bien du projet et sa lisibilité, plusieurs normes ont été mises en place. En voici la liste exhaustive:

- Pep 8 - python 3
- Auto code formatting - flutter
- Google JavaScript Style Guide - JavaScript
- GNOME - C

## Business Model & Plan

<b><u>partenaire:</u></b> - constructeur hardware - magasin de vente d'instrument de musique - studio - centre de recherche sur l'audio / musique	<b><u>Activités clés:</u></b> -développement continue de la solution -negocier la fabrication - marketing	<b><u>Proposition de valeur:</u></b> -synthétiseur d'accompagnement musicale interactif  -enregistrement et partage de création  revendeur: - mise en place d'un lien pour la revente de la solution	<b><u>relation client:</u></b> -site web, instagram, linkedin, api public -application mobile	<b><u>Segments de marchés:</u></b> -Client achetant la solution  -Simple utilisateur de l'application mobile
<b><u>structure de couts:</u></b> -salaire -construction boitier -location bureau -livraison client -serveur / production	<b><u>Source de revenu:</u></b> -prix du boitier -achat de nouveaux instrument digitalisé sur le boitier			

Nous développons une boîte physique dans nos locaux permettant de lire et écrire de la musique en format MIDI. Nous proposons donc aux musiciens de loisirs et pros un synthétiseur d'accompagnement musical permettant l'enregistrement et le partage de leurs créations.

Toute l'équipe de Jambox Company a un attrait particulier à la musique dite classique ou par ordinateur et périphériques divers et nous souhaitons retranscrire cette passion commune à travers notre produit. En ces temps de confinement social, la Jambox permet de se connecter aux autres et de partager ses œuvres facilement.

Les musiciens intéressés trouveront la boîte en vente chez les constructeurs directs, les points de vente de musique ou les studios d'enregistrement.

La promotion quant à elle sera effectuée par internet et sur nos réseaux sociaux ainsi que par notre application mobile et notre site.

## Stratégie de communication

Voici quelques aperçus des réseaux sociaux choisis pour notre communication et de nos premières publications lors de notre participation au DevFest de Nantes 2021 !

Instagram : <https://www.instagram.com/jamboxcompany/>



LinkedIn : <https://www.linkedin.com/company/jambox-company>

A screenshot of a LinkedIn post from the company page "Jambox Company". The post starts with a message: "Nous venons de créer notre page. Consultez notre page pour vous tenir au courant des dernières nouvelles de Jambox Company !". It includes a link to their LinkedIn profile: "Jambox Company | LinkedIn" and "fr.linkedin.com • Lecture de 1 min". Below the message are standard LinkedIn interaction buttons: "J'aime", "Commenter", "Partager", and "Envoyer".



Site internet :





## La Jambox (IOT)

Nous avons choisi la solution de la machine virtuelle avec VirtualBox pour répondre aux attentes de notre analyse. Le but de cette analyse est de vérifier si l'exécution de la librairie MIDO ne demande pas trop de ressources sur un Hardware de ce type.

Cette analyse étant lancée de la façon suivante:

- Exécution d'un script python avec une seule fonction de la librairie MIDO
- Exécution d'un script bash analysant la consommation de CPU et de mémoire en fonction d'un Process ID (pid) une seconde avant et une seconde après l'exécution du script python.

Les fonctions de librairies MIDO choisies sont des fonctions permettant l'extraction ou la modification des des fichiers midi ou tout simplement la lecture d'information du fichier.

Nous avons remarqué que les fonctions primaires de la librairie MIDO n'utilisent que très peu de ressources, à savoir moins de 2% de CPU et 0.4% de la mémoire de l'émulateur. Nous pouvons en conclure que l'utilisation du raspberry pi nous permettra d'utiliser le software sans inconvenient de ressources.

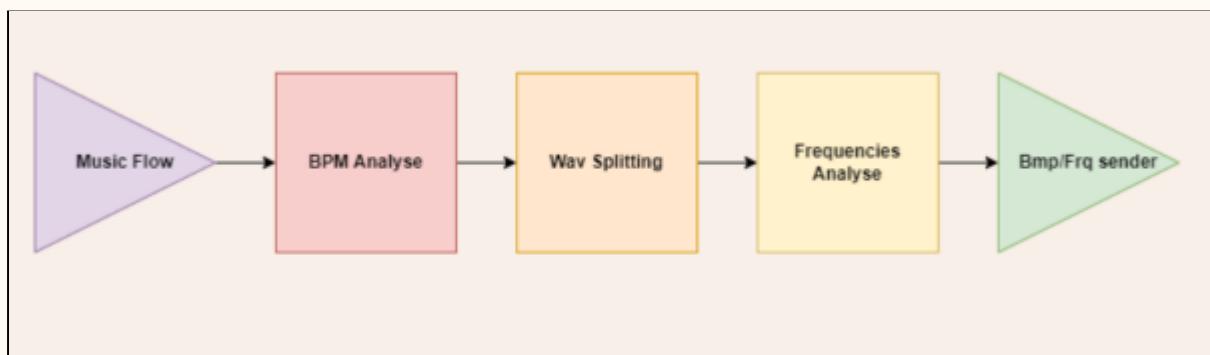
## IA - Lecture et génération de musique

### Algorithme de Récupération des notes/tempo :

Le but de l'algorithme est de récupérer des informations d'un fichier wav en entrée et d'ensuite fournir ces informations vers la partie IA du projet. Les informations récupérées du fichier sont le tempo et la fréquence.

Pour récupérer le tempo et la fréquence, différentes parties et opérations mathématiques sont implémentées :

- Analyse du BPM en utilisant les amplitudes du fichier wav
- Segmentation du fichier wav en fonction de différentes amplitudes analysées pour de meilleur performances
- Récupération des fréquences fondamentales grâce aux BPM et à des transformation de Fourier
- Envoie des fréquences/BMPs à l'algorithme de génération de midi



Toutes les parties de l'algorithme ont été réalisées mais plusieurs faiblesses ont été remarquées.

La précision des fréquences et du BPM obtenus peuvent varier en fonction des parasites enregistrés en même temps que la musique. Ce phénomène est accentué car l'instrument visé par l'algorithme est une guitare et lors d'un accord plusieurs notes sont jouées en même temps et donc plusieurs cordes vibrent avec des intensités variables. De plus, si l'utilisateur fait des erreurs, même minimes, elles seront très vite répercutées dans l'analyse du sample de musique.

Devant ces résultats et une difficulté à réaliser des tests réalistes, nous avons choisi d'abandonner cette partie. Bien que l'algorithme soit fonctionnel, son utilisation en condition réelle a une précision trop inconstante.

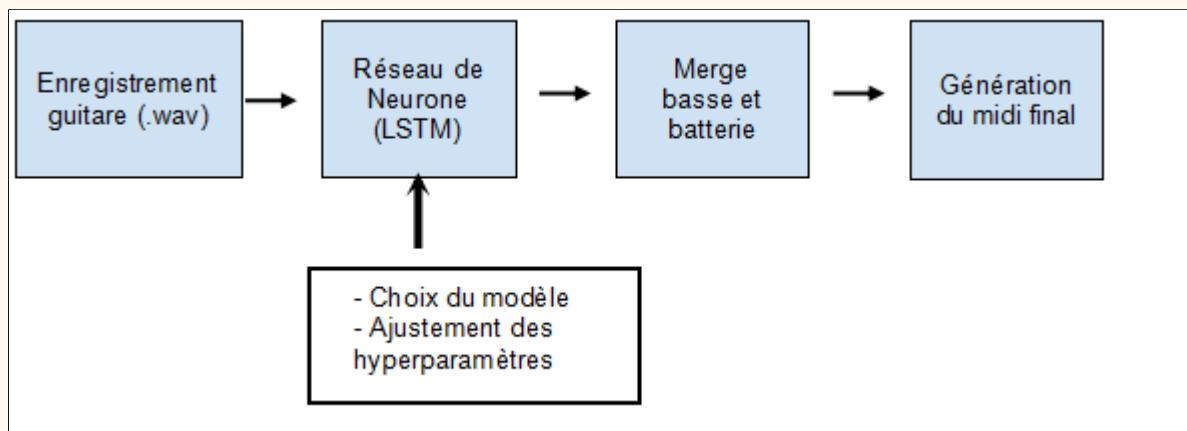
Il a été décidé que la meilleure solution était d'utiliser une carte son directement branchée soit à l'ampli soit à la guitare pour récupérer directement les informations, augmentant drastiquement les performances de la Jambox. Cette solution pose de même un autre problème : celui de l'augmentation du coût du produit. En effet le prix d'une carte son ainsi que celui de la licence d'un logiciel tiers pouvant récupérer la fréquence/Bpm ou même directement le Midi de la musique jouée augmentent le prix de la Jambox.

## Réseau de neurones LTSM (Long Term Short Memory) de génération de notes :

Le but de l'algorithme est de pouvoir générer de la batterie grâce à une intelligence artificielle en se basant sur un fichier audio d'une guitare ou d'un autre instrument.

Plus précisément, l'algorithme se base sur un fichier avec l'extension .wav comportant l'audio de la guitare et en ressort un fichier avec l'extension .mid comprenant la batterie générée artificiellement ainsi que la ligne de basse.

La génération de la batterie est réalisée via un réseau de neurones LSTM, basé sur la librairie magenta de Google.



Dans un premier temps la guitare est enregistrée en format wav. Le fichier wav est ensuite envoyé dans la réseau de neurones qui va générer plusieurs pistes de batterie en fonction des différents modèles. L'utilisateur aura donc le choix entre plusieurs pistes de batterie.

La troisième étape importante du processus est le merge des pistes de batterie et de la piste de basse. La piste de basse est générée avec un algorithme générique qui va récupérer le rythme de la piste de batterie ainsi que les différentes notes de la guitare de

base. Si les différentes notes de la guitare ne sont pas accessibles la basse sera générée en fonction des notes de la batterie.

Le processus global fonctionne dans un docker afin de prévenir des problèmes de version qui doivent être spécifiquement installés pour garantir le fonctionnement des différentes librairies.

L'architecture est la suivante:

- Dockerfile
- config.py
- magenta\_drums.py
- main.py
- mido\_lib.py
- requirements.txt
- test.py

Données :

- E\_Gallop\_Riff\_1a.wav
- bbox.wav clap.wav
- g\_t\_120\_50.wav
- loop.wav piano.wav
- sans\_titre.wav

Modèles (/models) :

- cat-drums\_2bar\_small.hikl.tar
- groovae\_2bar\_humanize.tar
- groovae\_4bar.tar
- groovae\_2bar\_add\_closed\_hh.tar
- groovae\_2bar\_tap\_fixed\_velocity.tar
- nade-drums\_2bar\_full.tar

- Les fichiers python .py contiennent les différents algorithmes permettant la génération.
- Quant au fichier ‘config.py’, il contient les différentes traductions des normes midi ainsi que celles pour les conversions de fréquence.
- Le fichier ‘magenta\_drums.py’ permet la génération de la batterie via le réseau de neurones.
- Le fichier ‘mido\_lib.py’ permettant le process sur les fichiers midi.
- Le ‘main.py’ appelle les fonctions permettant de relier tous les processus de génération.

Le dossier data correspond aux fichiers au format .wav permettant le test du processus. Pour générer la batterie d'un nouveau fichier audio il faudra positionner le fichier dans ce dossier.

Le dossier Modèles comprend les modèles pré entraînés permettant la génération de la batterie.

Pour lancer le processus il faut dans un premier temps positionner son fichier d'entrée au format wav dans le dossier data.

Le docker peut ensuite être lancé grâce à ces commandes:

```
$ sudo docker build -t magenta .
```

```
$ sudo docker run -it magenta /bin/bash
```

Une fois dans le docker, il suffit de lancer le fichier main avec en argument le nom du fichier se trouvant dans data avec la commande '*python3 main.py \$wavname.wav*'.

Si aucun fichier n'est passé en argument le réseau de neurones générera un fichier random avec une batterie et une basse en fonction d'un tempo pouvant être modifié.

À la fin du processus, plusieurs fichiers sont générés, les fichiers midi contenant la batterie pour chaque modèle, les fichiers midi combinant la batterie et la basse pour chaque modèle et les fichiers .wav combinant la batterie, la basse et l'entrée de base de la guitare ayant servi à générer la sortie.

## API REST

Afin de développer le backend, nous avons choisi Node.js pour la liberté qu'il offre et qui nous a permis d'explorer des fonctionnalités qui nous ont aidé dans le projet. La base de données est une MongoDB, permettant de gérer des données complexes comme les notes ou accords de musiques aux multiples formats.

Cette API permet trois actions principales :

- L'inscription d'un utilisateur
- la connexion/déconnexion d'un utilisateur
- L'inscription et la connexion d'un utilisateur par son identifiant google (Oauth2)

Voici les routes de notre swagger :

- POST Sign Up :
  - <http://petstore.swagger.io/v1/api/auth/signup>
  - Permet de se connecter
- POST Sign In :
  - <http://petstore.swagger.io/v1/api/auth/signin>
  - Permet de s'inscrire
- POST Refresh Token :
  - <http://petstore.swagger.io/v1/api/auth/token/refresh>
  - Permet de rafraîchir le token d'authentification
- POST Sign Out :
  - <http://petstore.swagger.io/v1/api/auth/signout>
  - Permet de se déconnecter
- GET Get User Profile
  - <http://petstore.swagger.io/v1/api/profile>
  - Permet de récupérer le profil utilisateur une fois connecté à l'appli : Les infos des précédentes sessions de Jam', les réglages pré-enregistrés par exemple

## Application mobile

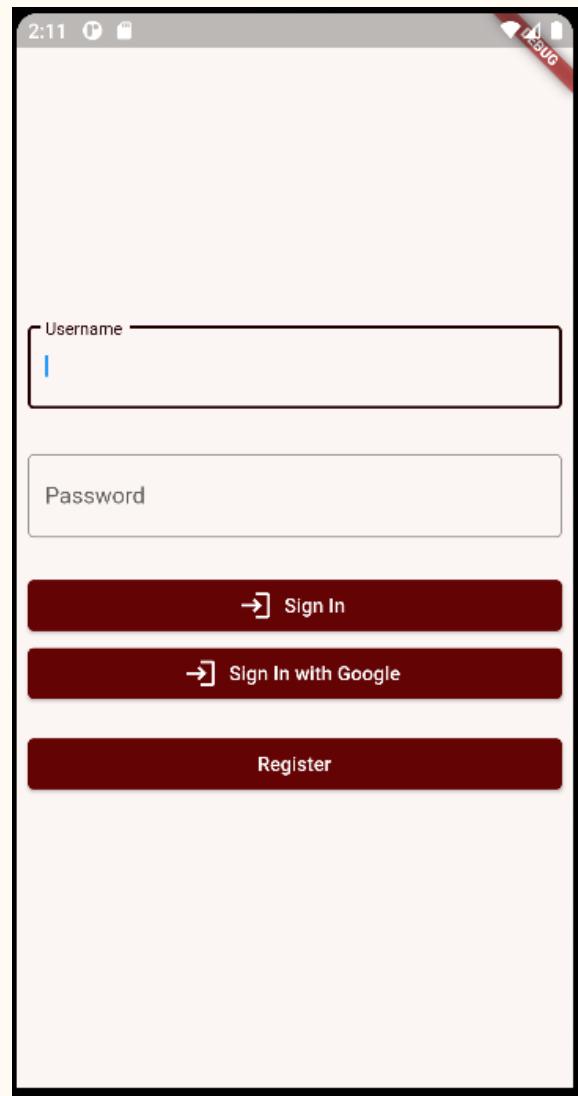
Nous avons choisi d'utiliser Flutter pour ses performances et sa meilleure adaptabilité à toutes les versions d'Android et IOS. Flutter également compense son manque de communauté par une excellente documentation

L'écran de connexion permet à l'utilisateur de s'authentifier. L'application n'est pas utilisable sans que l'utilisateur ne se connecte.

L'utilisateur peut :

- Créer un compte
- Se connecter
- Se connecter via Google (Oauth)

Dans tous les cas l'utilisateur est enregistré dans la base de données via l'API.

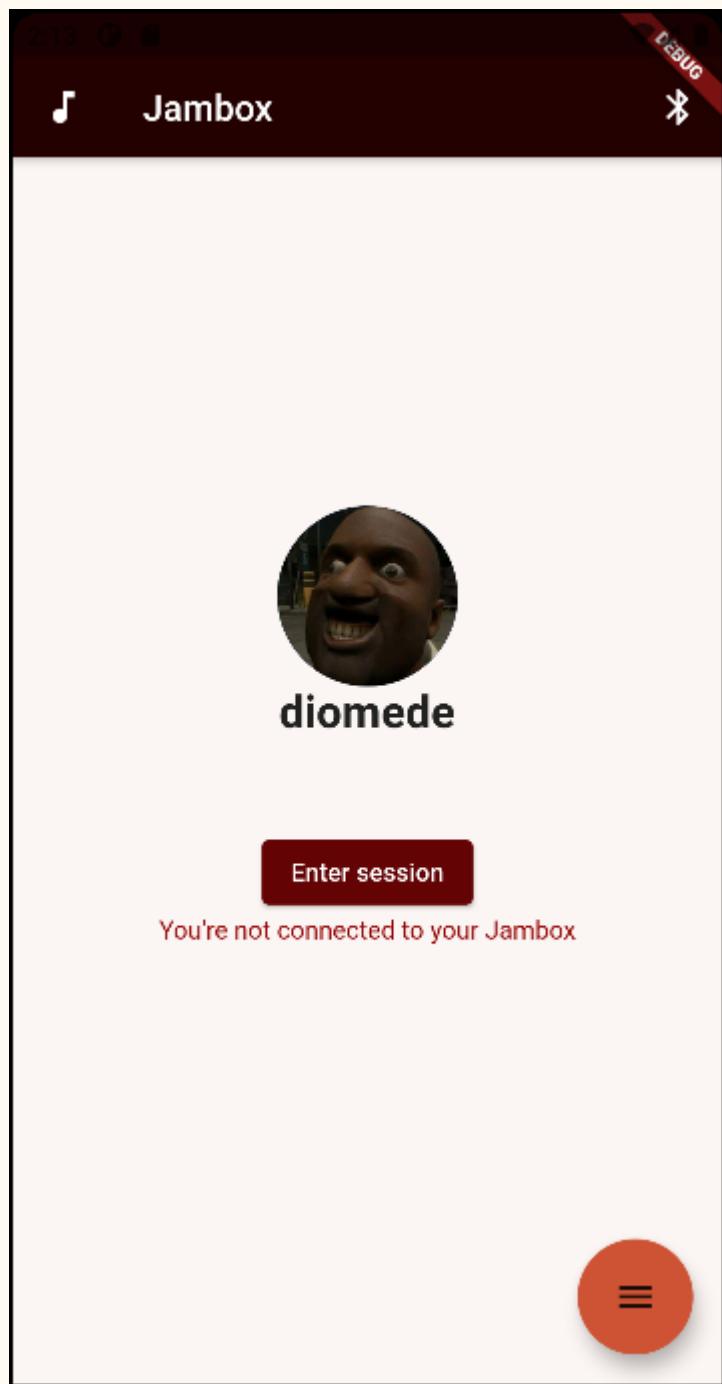


L'écran principal permet d'accéder aux différents menus.

L'utilisateur peut :

- Cherche sa box (Bluetooth)
- Commencer une session
- Accéder aux paramètres
- Accéder aux enregistrements

Cette page pourrait aussi servir de réseau social pour partager leurs enregistrements avec d'autres clients.

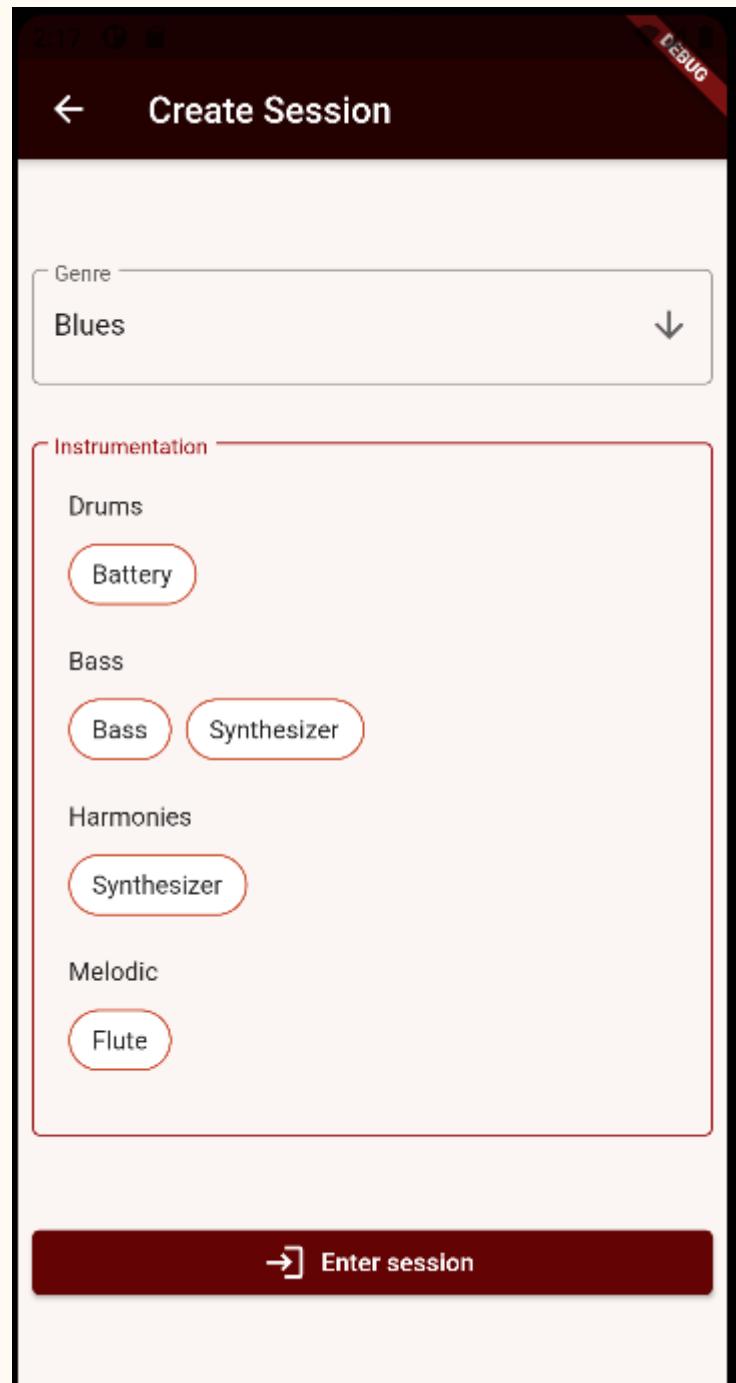


L'écran de création de session permet de configurer les variables d'entrées de l'algorithme.

L'utilisateur peut :

- Choisir le type de musique
- Choisir l'instrumentation

Cette page évoluerait en fonction des variables d'entrées possibles de l'algorithme.

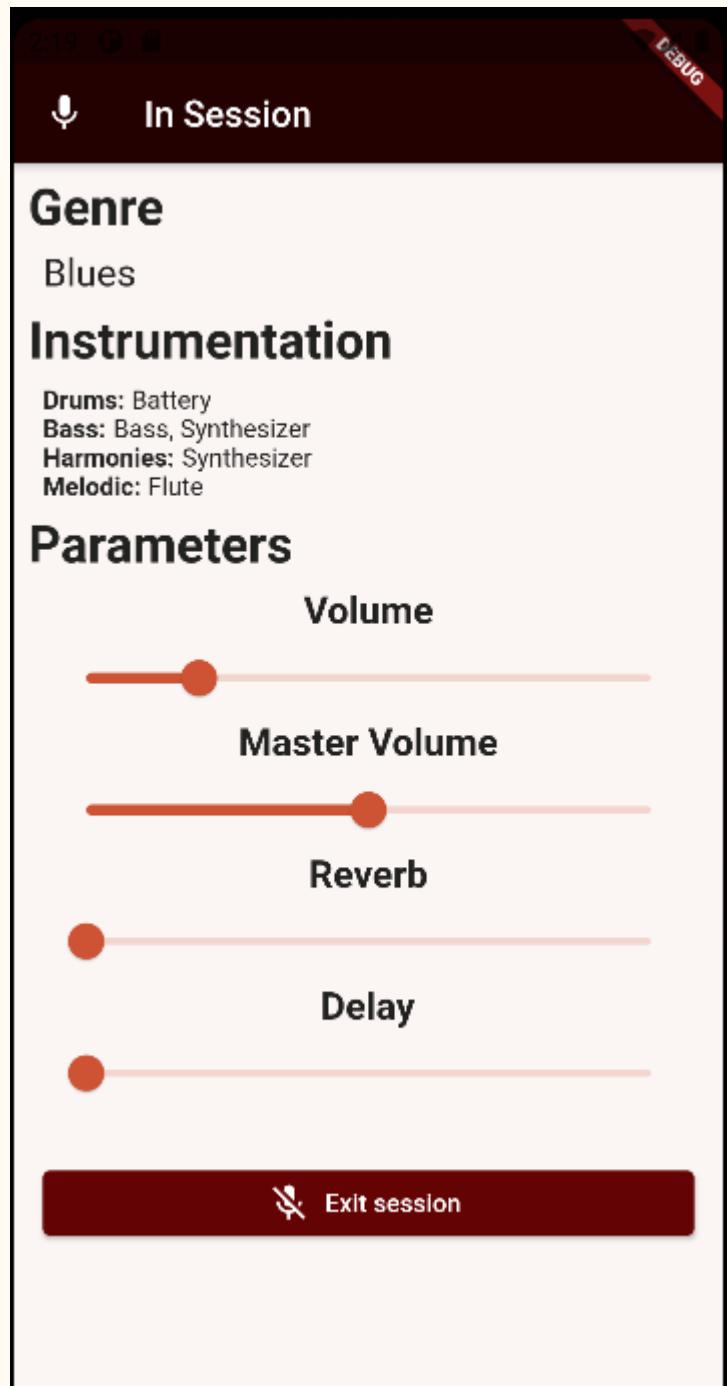


Pendant une session l'utilisateur peut jouer avec l'intelligence artificielle et changer les paramètres de sortie de l'algorithme.

L'utilisateur peut :

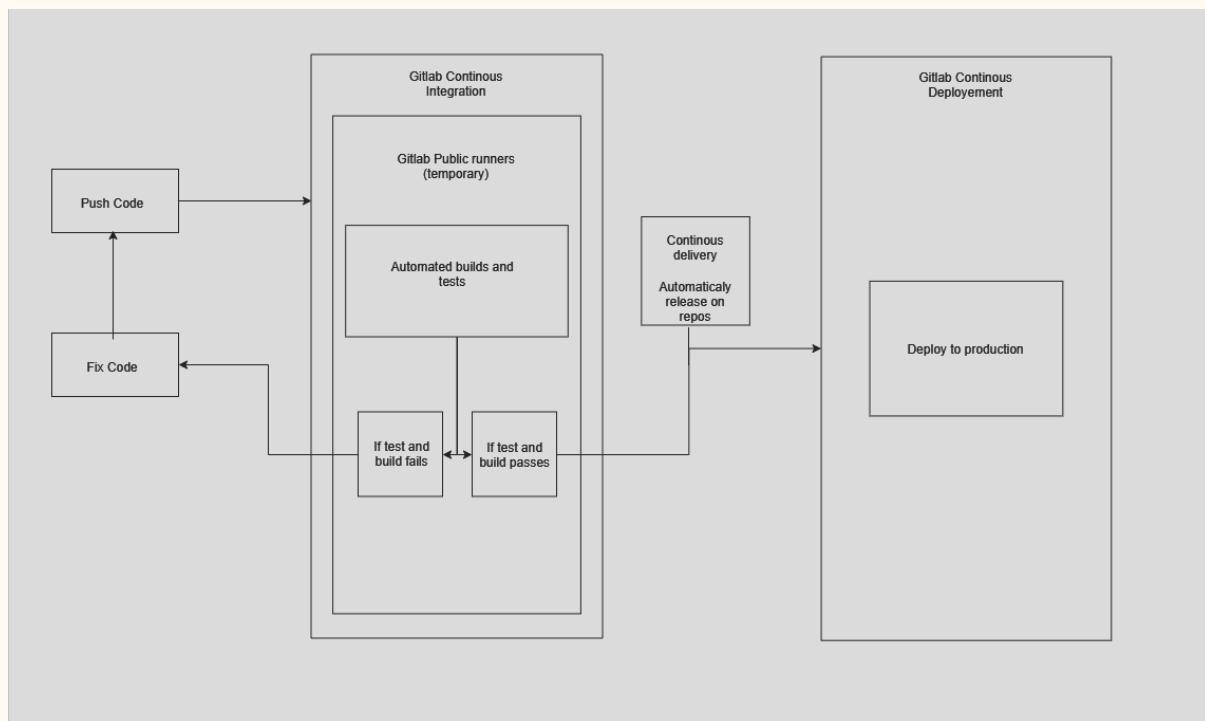
- Moduler le volume global
- Moduler le volume d'entrée
- Utiliser des effets intégrés

Nous pourrions aussi ajouter une modulation de volume par instrument ou encore plus d'effets intégrés à la box.



## Déploiement

Afin d'intégrer, de tester et de déployer notre code sur un dépôt Git partagé, nous avons choisis de mettre en place un serveur GitLab. Facile et rapide d'utilisation, reposant sur l'écriture de fichier Yaml, cette solution open-source correspondant aux besoins de notre projet



Nos pipelines fonctionnent selon les normes généralement appliquées aux entreprises, à chaque push (donc merge), nos Runners construisent notre application et lancent les tests associés. A la fin de ces derniers, en fonction de la validité ou non, soit on corrige le code soit les tests sont réussis et ils sont mis en package, puis déployé en production.

## Pour conclure

Après plus de 14 mois de travail, la réalisation de ce projet se termine. Cette expérience se termine avec un goût amer du fait de la réalisation incomplète du produit due aux problèmes encourus le long de ce voyage.

Ce fut tout de même un travail collectif extrêmement intéressant tant sur le plan technique qu'humain, introduisant de nouveaux challenges jusque-là jamais rencontrés pendant nos études : un projet entier, de la création de concept au rendu commercial, de la création de l'équipe à son évolution finale sur le long terme.

Nous avons dû nous adapter aux limites de connaissances, techniques et évoluer à partir de celles-ci ainsi que gérer notre temps sur de grandes périodes de travail ponctuées par les autres projets en cours, reproduisant une vie d'entreprise miniature. Notre passion commune pour la musique nous a poussé à donner le meilleur de nous mêmes malgré les embûches, et nous sommes heureux d'avoir mené à bout ce projet tous ensemble.

## Sources

- <https://developer.gnome.org/programming-guidelines/stable/c-coding-style.html.en>
- <https://google.github.io/styleguide/jsguide.html>
- <https://flutter.dev/docs/development/tools/formatting>
- <https://www.python.org/dev/peps/pep-0008/>