# Synthétiseur de rêves – Dossier de cadrage (version locale)

# Contexte

Le projet Synthétiseur de rêves a évolué : l’application fonctionne désormais entièrement en local dans VS Code, sans Jupyter. Les utilisateurs lancent streamlit run app.py, les clés API sont stockées dans un fichier .env, et les données personnelles (dreams.json) restent sur la machine. Les appels aux API (Groq Whisper, Mistral, Clipdrop) se font toujours via Internet mais aucune donnée n’est stockée côté serveur.

# Objectifs

• Transcrire un rêve audio (.wav/.mp3/.m4a)

• Détecter et pondérer 6 émotions clés (heureux, anxieux, triste, en\_colere, fatigue, apeure)

• Générer une image onirique à partir du texte

• Sauvegarder chaque rêve dans un tableau de bord local

• Fournir une UX simple : 3 onglets (Synthétiseur, Tableau de bord, Aide)

# Stack & environnement local

|  |  |  |
| --- | --- | --- |
| Composant | Usage | Détails |
| Python ≥ 3.10 | Langage principal | Virtualenv venv/ recommandé |
| Streamlit | UI temps réel | pip install streamlit |
| python‑dotenv | Sécurité clés | .env ➔ variables d’environnement |
| Groq + Whisper (v3‑turbo) | ASR (transcription audio) | Latence 2‑5 s, modèle multilingue |
| Mistral small | Analyse émotionnelle (LLM) | sortie JSON stricte |
| Clipdrop Text‑to‑Image | Génération d’images | Stable Diffusion XL tuned |
| JSON local | Persistance | dreams.json (append) |

# 4. Architecture locale

L'utilisateur interagit via Streamlit : audio -> transcription (Groq) -> émotions (Mistral) -> image (Clipdrop) -> sauvegarde locale dans dreams.json.

# 5. Benchmark détaillé des APIs (tarifs & quotas 2025-07-04)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| API | Offre gratuite | Tarif au‑delà | Latence (moy.) | RGPD / hébergement | Avantages | Limites |
| Groq Whisper v3‑turbo | 120 min/mois (≈ ~500 requêtes courtes) | **0,00015 $/seconde** (≈ 0,54 $/h audio) | 2‑5 s | 🇪🇺 (Frankfurt) – données éphémères | Modèle SOTA, multilingue, peu coûteux | Pas de batch upload ; audio envoyé hors machine |
| Mistral small | 6 M tokens/mois | **2 $/1 M tokens** ensuite | 2‑3 s | 🇪🇺 (Paris/Frankfurt) | RGPD‑compliant, rapide | Pas encore de modèle émotion « finetuned » dédié |
| Clipdrop T2I | 100 images/jour **ou** 1 000 cr/mois free | **10 $/1 000 images** (≈ 0,01 $/img) | 4‑8 s | 🇺🇸 / 🇪🇺 mix (AWS) | SD‑XL de bonne qualité, rapide | Couleurs parfois fantaisistes, max 1024×1024 |
| Alternatives | **OpenAI Whisper** (5 h free) | 0,006 $/min |  |  | + fichiers >25 Mo ok | Stockage US par défaut |
| **AssemblyAI** (30 min free) | 0,015 $/min | 3‑6 s | 🇺🇸 | support diarization | tarif 2‑3× Groq |
| **OpenAI DALL·E 3** | – | 0,04 $/img | 10‑15 s | 🇺🇸 | images + cohérentes texte  coût ×4 Clipdrop |
| **Stability DreamStudio** | 25 cr free | 0,01 $/img | 4‑7 s | 🇪🇺 | open‑weights  nécessité crédit upfront |

**Coût mensuel estimatif (scénario 30 rêves / mois)**

|  |  |  |  |
| --- | --- | --- | --- |
| Ressource | Volume | Coût unité | Montant |
| Transcription | 30 min audio | 0,54 $/h → **0,27 $** | ≈ 0,27 $ |
| Analyse émotionnelle | 60 000 tokens | 2 $/1 M → **0,12 $** | ≈ 0,12 $ |
| Génération image | 30 images | 0,01 $/img → **0,30 $** | ≈ 0,30 $ |
| Total mensuel |  |  | **≈ 0,69 $** |

*Moins d’1 € par mois pour un usage personnel de 1 rêve/jour.*

# 6. Déploiement local

1. Cloner le dépôt GitHub : https://github.com/AdrienAkilal/Synthese-reves

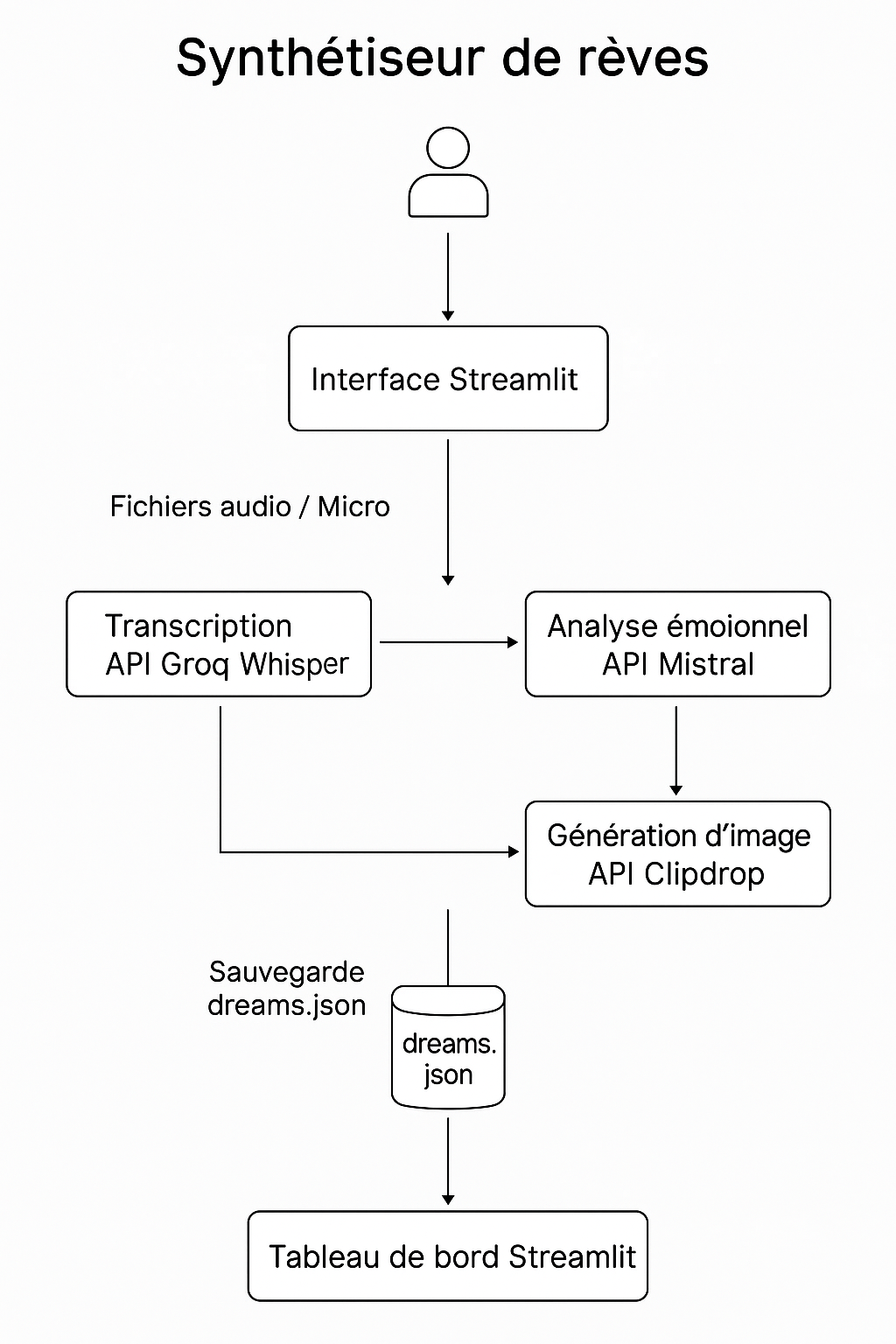
2. Créer l’environnement : python -m venv venv && source venv/bin/activate

3. Installer les dépendances : pip install -r requirements.txt

4. Créer le fichier .env avec les clés API

5. Lancer l’app : streamlit run app.pyDiagramme d’architecture

**7. Architecture de l’application**

****

Voici une **description synthétique** du diagramme d’architecture du projet **Synthétiseur de rêves** :

**Composants clés :**

* **Utilisateur**  
  Interagit via une interface Streamlit locale dans son navigateur.
* **App Streamlit (app.py)**  
  Point d’entrée de l’application. Gère la navigation et appelle les fonctions de traitement.
* **Fichier .env**  
  Stocke les clés API nécessaires pour interagir avec les services externes.
* **Fichier dreams.json**  
  Contient localement les rêves sauvegardés (texte, émotions, image).
* **APIs externes**
  + **Groq (Whisper)** → Transcription de l’audio en texte
  + **Mistral** → Analyse des émotions
  + **Clipdrop** → Génération d’image onirique

**Flux général :**

1. L'utilisateur charge un audio.
2. L’app transcrit (Groq), analyse (Mistral), génère une image (Clipdrop).
3. Le tout est sauvegardé localement dans dreams.json.

# 7. Diagramme montrant le fonctionnement de l’application (diagramme de séquence)

**Acteurs et objets :**

* **Utilisateur (acteur)**
* **Interface Streamlit**
* **App**
* **Groq API**
* **Mistral API**
* **Clipdrop API**
* **Fichier JSON (dreams.json)**

**Synthèse complète d’un rêve**

**1. L’utilisateur lance l’app Streamlit**

* Il choisit d’enregistrer ou d’uploader un fichier audio.

**2. Streamlit affiche l’interface**

* Il envoie l’audio sélectionné à l’**App**.

**3. App appelle Groq API**

* La méthode transcribe() envoie l’audio à Groq Whisper.
* La transcription textuelle est retournée.

**4. App appelle Mistral API**

* La méthode analyse\_emotion() envoie le texte transcrit à Mistral.
* Elle reçoit une prédiction brute JSON des émotions.
* Elle applique un softmax pour lisser les scores.

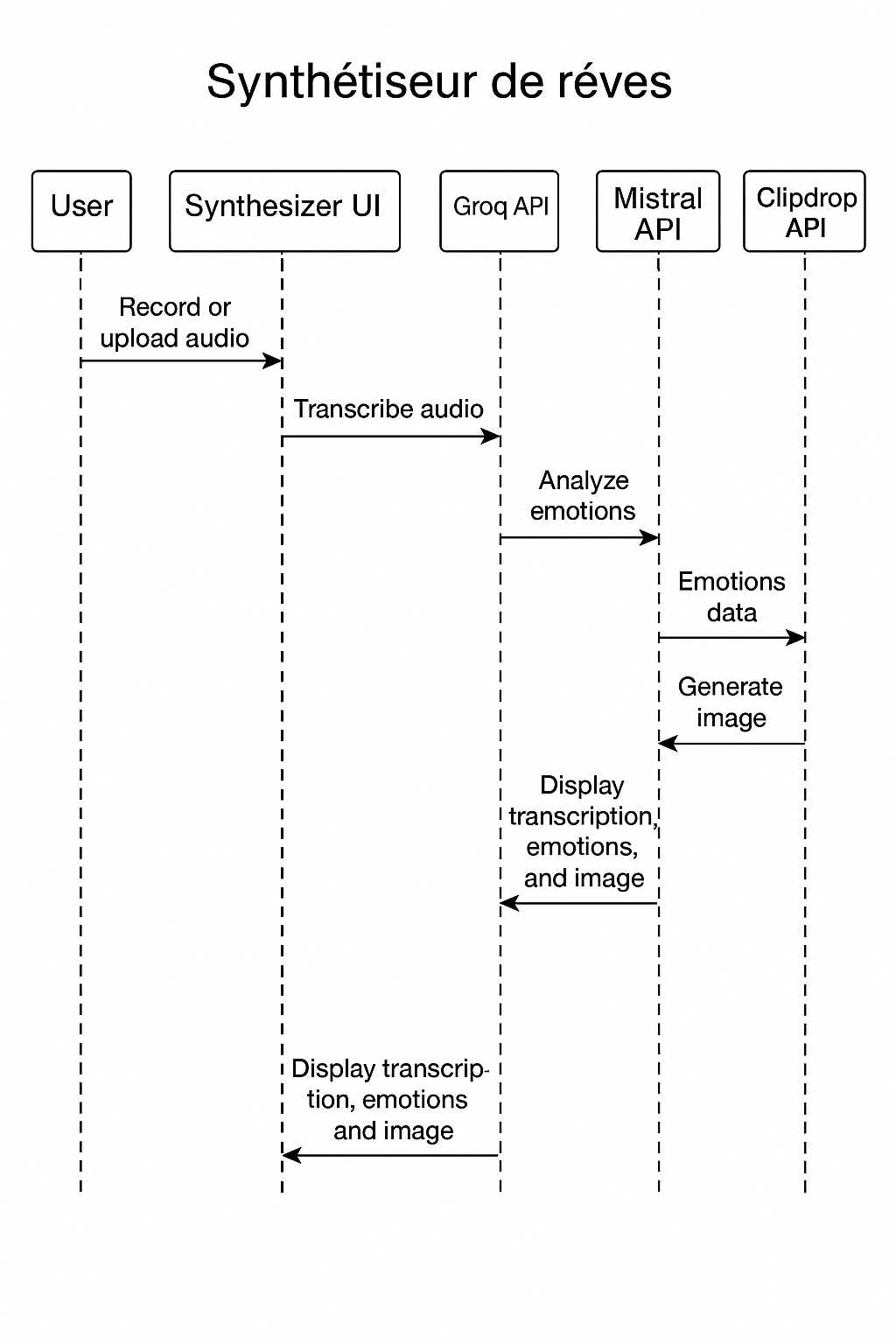
**5. App appelle Clipdrop API**

* La méthode generate\_image() envoie le texte à Clipdrop.
* Elle reçoit une image générée (format binaire).

**6. App appelle save\_dream()**

* Les données du rêve (texte, émotions, image encodée) sont encodées et stockées dans dreams.json.

**7. Streamlit affiche le résultat**

* Texte transcrit, émotions pondérées, image générée et confirmation de sauvegarde.

**Ce que montre le diagramme :**

* Les **interactions enchaînées** de manière synchrone
* Les **temps de réponse** visibles par les flèches descendantes/montantes
* L’**ordre exact des traitements**
* Le rôle de chaque API et des méthodes internes