# 📘 Documentation – API Flask : Pipeline de Reconnaissance Faciale (Modulaire)

## 1. Présentation

Cette API Flask exécute un pipeline de traitement d'images de visages. Chaque image est associée à un sujet identifié dans le nom de fichier. Le pipeline traite les images sujet par sujet en appliquant les étapes suivantes : prétraitement, réduction de dimension par PCA (eigenfaces), ajout de bruit différentiel, et reconstruction.

## 2. Format attendu des fichiers

Les fichiers doivent être nommés selon ce format :

subject\_<A>\_<B>\_<description>.png

- <A> : Identifiant du sujet  
- <B> : Numéro de l'image  
- <description> : texte libre

## 3. Lancement de l'application

Lancer le serveur en exécutant le script principal :

python src/modules/face\_processing\_api.py

## 4. Endpoints de l'API

### 4.1 POST /pipeline

Lance le pipeline complet sur un dossier d’images.

Paramètres :

- folder : chemin vers le dossier contenant les images (obligatoire)

- epsilon : paramètre de bruit différentiel (optionnel, par défaut : 9.0)

- n\_components\_ratio : ratio de composantes PCA (optionnel, par défaut : 0.9)

Exemple :

curl -X POST http://localhost:5000/pipeline \  
-F "folder=data/database" \  
-F "epsilon=7.5" \  
-F "n\_components\_ratio=0.85"

### 4.2 GET /result/<subject>/<step>

Récupère une étape du pipeline pour un sujet donné.

Exemple :

curl http://localhost:5000/result/1/eigenfaces

Étapes disponibles :

- resized  
- grayscale  
- normalized  
- flattened  
- eigenfaces  
- mean\_face  
- projection  
- noised\_projection  
- reconstructed

## 5. Sauvegarde automatique

Le premier sujet (selon l'ordre alphabétique) traité est automatiquement exporté en images JPEG dans le dossier `show\_test\_subject/`.

## 6. Logs

Tous les détails du traitement sont enregistrés dans : `logs/face\_pipeline\_debug.log`. Ce fichier contient les erreurs, informations de progression, paramètres utilisés, etc.

## 7. Fonctionnement modulaire

Le traitement est découpé en fonctions indépendantes :

- preprocess\_image : redimensionnement, grayscale, normalisation, flattening

- compute\_eigenfaces : calcul de la PCA et extraction des eigenfaces + image moyenne

- add\_differential\_noise : ajout de bruit Laplacien sur les projections

- reconstruct\_images : reconstruction depuis les projections bruitées

- pil\_to\_base64 / array\_to\_base64 : utilitaires pour affichage ou API

## 8. Recommandations

- Avoir au moins 2 images par sujet pour permettre la PCA

- Nettoyer le dossier `show\_test\_subject/` si besoin avant relance

- Fournir des images de taille et format homogènes

## 9. Remarques

Ce pipeline est idéal pour l'expérimentation en traitement d'images, apprentissage de la PCA et protection des données personnelles. Il peut être utilisé comme base pour des extensions : visualisation, comparaison, ou même intégration front-end.