

Projet de Cloud & Edge Computing

Hébergement d'une application d'indexation et de recherche multimédia

Nicolas Melaerts et Kenza Khemar
Université de Mons
Master 1 en Informatique

17 juin 2025

Table des matières

Introduction	3
1 Hébergement de l'application sur le Cloud	3
1.1 Configuration de la machine virtuelle	3
1.2 Configuration de la machine virtuelle	3
2 Déploiement et Infrastructure Docker	3
2.1 Scripts de déploiement	3
2.2 Fichier docker-compose.yml	4
2.3 Dockerfile du service SaaS	5
3 Nginx	5
3.1 Rôle de Nginx	5
3.2 Configuration globale de Nginx	6
4 Résumé des ports utilisés	7
4.1 Cybersécurité	7
5 Indexation et Recherche Multimédia	8
5.1 Descripteurs utilisés	8
5.2 Méthodes de recherche et de similarité	8
5.3 Affichage des résultats	8
5.4 Évaluation des performances	9
A Exemples de résultats visuels (Interface et Résultats)	10
A.1 Page de connection	10
A.2 Page d'accueil	10
A.3 Menu 1 : Affichage des images 1	11
A.4 Menu 1 : Calcul des descripteurs 1	11
A.5 Menu 3 : Recherche par descripteurs 1	12
A.6 Menu 3 : Recherche par descripteurs 2	12
A.7 Menu 4 : Recherche par texte 1	13
A.8 Menu 4 : Recherche par texte 2	13
A.9 Menu 5 : Recherche Deep Learning 1	14
A.10 Menu 5 : Recherche Deep Learning 2	14
A.11 Menu 6 : Génération Vasarely 1	15
A.12 Menu 6 : Génération Vasarely 2	15

Introduction

Ce rapport présente le projet réalisé dans le cadre de l'UE « Cloud & Edge Computing » (2024-2025), portant sur l'hébergement d'une application d'indexation et de recherche multimédia sur une ressource Cloud.

1 Hébergement de l'application sur le Cloud

1.1 Configuration de la machine virtuelle

L'application a été déployée sur une machine virtuelle Linux sous Ubuntu 24.04.2 LTS (nom de code « noble »).

La machine dispose des ressources suivantes :

- Mémoire vive totale : 3,8 Go, dont environ 2,3 Go disponibles au moment du test.
- Disque principal : 48 Go au total, avec 36 Go utilisés et 11 Go libres.

Ces caractéristiques sont suffisantes pour exécuter l'application et ses services Docker dans des conditions stables.

1.2 Configuration de la machine virtuelle

Après création, la machine a été configurée manuellement avec les étapes suivantes :

- Mise à jour du système via `sudo apt update && sudo apt upgrade`
- Installation de Docker : ajout du dépôt officiel et installation via APT
- Installation de Docker Compose
- Clonage du dépôt Git contenant l'application
- Lancement de l'environnement via `download_and_unzip.sh && deploy_server.sh`

2 Déploiement et Infrastructure Docker

2.1 Scripts de déploiement

Déploiement local : `deploy_local.sh`

```
#!/bin/bash
docker compose down          # Arrête et supprime les conteneurs
docker compose rm -f         # Force la suppression des conteneurs arrêtés
docker compose build         # Recrée les images Docker
docker compose up             # Démarrre les conteneurs en mode interactif
```

Déploiement serveur : `deploy_server.sh`

```
#!/bin/bash
echo " Pulling latest code..."
git pull origin main
echo " Stopping and removing old containers..."
docker-compose down
echo " Rebuilding images..."
docker-compose build
echo " Starting containers..."
docker-compose up -d
echo " Checking container status..."
docker-compose ps
echo " Deployment complete!"
```

2.2 Fichier docker-compose.yml

Ce fichier définit les services à déployer, leur configuration, et la manière dont ils interagissent.

Version et réseau

```
networks:  
  app-network:  
    driver: bridge
```

Services

nginx Reverse proxy HTTP :

```
image: nginx:latest  
ports:  
  - "80:80"  
volumes:  
  - ./nginx/nginx.conf:/etc/nginx/nginx.conf  
  - ./nginx/conf.d:/etc/nginx/conf.d  
depends_on:  
  - web  
networks:  
  - app-network  
restart: unless-stopped
```

web Application Flask :

```
build:  
  context: .  
  dockerfile: SaaS/Dockerfile  
expose:  
  - "8080"  
environment:  
  - FLASK_APP=app.py  
  - FLASK_ENV=development  
  - VASARELY_SERVICE_URL=http://vasarely:5002  
  - PYTHONPATH=/app:/opt/DESKTOP_APP  
depends_on:  
  - vasarely  
volumes:  
  - ./SaaS:/app  
  - ./DESKTOP_APP:/opt/DESKTOP_APP  
networks:  
  - app-network  
restart: unless-stopped
```

vasarely Création de pavage style Vasarely :

```
build: ./vasarely_service  
expose:  
  - "5002"  
volumes:  
  - ./vasarely_service:/app
```

```

networks:
  - app-network
restart: unless-stopped
deploy:
  resources:
    limits:
      memory: 2G

```

2.3 Dockerfile du service SaaS

```

FROM coolsa/pyqt-designer:x64

RUN pip install --no-cache-dir \
    numpy \
    opencv-python \
    matplotlib \
    scikit-image \
    scikit-learn \
    sentence-transformers \
    gdown

WORKDIR /opt/SaaS

COPY SaaS/requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY SaaS /opt/SaaS

EXPOSE 8080

ENV FLASK_APP=app.py
ENV FLASK_ENV=production

CMD ln -sf /opt/DESKTOP_APP/descriptors.py /opt/SaaS/descriptors.py && python3 app.py

```

3 Nginx

3.1 Rôle de Nginx

Nginx est utilisé dans ce projet comme **reverse proxy**. Son rôle principal est de recevoir les requêtes HTTP provenant des utilisateurs (via un navigateur web, par exemple) et de les rediriger vers le bon service en interne, ici l'application Flask.

Cette architecture présente plusieurs avantages :

- Elle permet d'accéder à l'application via le port 80 standard (`http://localhost`) sans exposer directement le port interne utilisé par Flask (8080).
- Elle offre une couche supplémentaire de sécurité grâce à des en-têtes HTTP configurables.
- Elle centralise les logs, les erreurs et les règles de timeout.
- Elle permet d'évoluer facilement vers une architecture plus complexe (avec plusieurs services ou un équilibrage de charge).

Nginx agit donc comme une passerelle entre l'extérieur et les services internes du projet.

3.2 Configuration globale de Nginx

Le fichier `nginx.conf` principal configure le comportement global du serveur Nginx. Voici son contenu et son explication.

```
user    nginx;
worker_processes  auto;

error_log  /var/log/nginx/error.log notice;
pid        /var/run/nginx.pid;
```

Explication :

- `user nginx` : définit l'utilisateur système exécutant les processus Nginx.
- `worker_processes auto` : ajuste automatiquement le nombre de processus de travail selon les CPU disponibles.
- `error_log` : définit le fichier de journalisation des erreurs.
- `pid` : indique où stocker l'identifiant du processus principal.

```
events {
    worker_connections  1024;
}
```

Explication : fixe le nombre maximum de connexions simultanées par processus de travail.

```
http {
    include       /etc/nginx/mime.types;
    default_type application/octet-stream;
```

Explication :

- `include mime.types` : permet de servir les fichiers avec le bon type MIME.
- `default_type` : type MIME utilisé si aucun autre n'est reconnu.

```
log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                  '$status $body_bytes_sent "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"';
access_log  /var/log/nginx/access.log  main;
```

Explication : crée un format de journalisation détaillé et définit le fichier de log des accès.

```
sendfile     on;
keepalive_timeout  300;
```

Explication :

- `sendfile on` : active un envoi efficace des fichiers (via le kernel).
- `keepalive_timeout` : fixe à 300 secondes la durée d'inactivité avant fermeture de la connexion.

```
proxy_connect_timeout 300s;
proxy_send_timeout 300s;
proxy_read_timeout 300s;
send_timeout 300s;
client_body_timeout 300s;
client_header_timeout 300s;
fastcgi_read_timeout 300s;
```

Explication : ces valeurs de timeout étendues (300 secondes) assurent que les requêtes longues ne soient pas interrompues prématurément.

```
server_tokens off;
add_header X-Content-Type-Options nosniff;
add_header X-Frame-Options SAMEORIGIN;
add_header X-XSS-Protection "1; mode=block";
```

Explication :

- `server_tokens off` : masque la version exacte de Nginx dans les en-têtes.
- `nosniff` : empêche les navigateurs de deviner le type MIME.
- `SAMEORIGIN` : interdit l'intégration du site dans des iframes externes (protection clickjacking).
- `X-XSS-Protection` : active un filtre anti-XSS côté navigateur.

```
include /etc/nginx/conf.d/*.conf;
}
```

Explication : permet d'ajouter des configurations supplémentaires (comme les règles du reverse proxy) depuis le dossier `conf.d`.

4 Résumé des ports utilisés

Voici un tableau récapitulatif des ports utilisés dans l'infrastructure Docker :

Service	Port exposé	Description
<code>nginx</code>	80 (externe)	Port d'accès HTTP depuis le navigateur
<code>web</code>	8080 (interne)	Serveur Flask (API / Interface web)
<code>vasarely</code>	5002 (interne)	Service bonus de génération d'image

TABLE 1 – Résumé des ports utilisés par les services Docker

4.1 Cybersécurité

Plusieurs pratiques de sécurité ont été appliquées :

- **Headers HTTP sécurisés** ajoutés via Nginx :
 - `X-Content-Type-Options: nosniff`
 - `X-Frame-Options: SAMEORIGIN`
 - `X-XSS-Protection: "1; mode=block"`
- **Désactivation des tokens serveur** : `server_tokens off` masque la version exacte de Nginx.
- **Timeouts étendus** pour éviter les attaques par lenteur (`slowloris`, etc.).
- **Accès restreint au port 80** depuis l'extérieur, seuls les ports nécessaires sont ouverts.
- **Authentification utilisateur** gérée avec Flask-Login :
 - Utilisation d'une clé secrète pour la session Flask.
 - Implémentation d'un système simple d'authentification avec un utilisateur unique (à adapter en production).
 - Protection des routes avec le décorateur `@login_required`, sauf pour le service Vasa-rely qui reste accessible publiquement.
 - Gestion des redirections après login/logout pour une meilleure expérience utilisateur.
 - Hachage sécurisé des mots de passe avec `werkzeug.security.generate_password_hash` et vérification avec `check_password_hash`.

5 Indexation et Recherche Multimédia

5.1 Descripteurs utilisés

Pour l'indexation et la recherche d'images dans le service SaaS, plusieurs types de descripteurs sont utilisés, couvrant à la fois des méthodes classiques et des modèles profonds (deep learning) :

Descripteurs classiques :

- BGR : Histogrammes de couleurs dans l'espace BGR.
- HSV : Histogrammes de couleurs dans l'espace HSV.
- GLCM : Matrice de cooccurrence des niveaux de gris pour la texture.
- HOG : Histogram of Oriented Gradients pour la forme et la texture.
- LBP : Local Binary Patterns pour la texture.
- ORB : Oriented FAST and Rotated BRIEF pour les points d'intérêt.

Descripteurs textuel :

- all-MiniLM-L6-v2

Descripteurs profonds (Deep Learning) :

- GoogLeNet
- Inception v3
- ResNet
- Vision Transformer
- VGG

Ces modèles sont utilisés en mode « feature extractor » : on retire la dernière couche de classification et on utilise le vecteur de caractéristiques produit par la couche précédente.

5.2 Méthodes de recherche et de similarité

Le service SaaS propose plusieurs mesures de similarité/distance pour comparer les vecteurs de caractéristiques :

- Euclidean (Euclidienne) : Distance classique entre deux vecteurs.
- Manhattan : Somme des valeurs absolues des différences.
- Cosinus : Mesure l'angle entre deux vecteurs (similitude cosinus).
- Correlation : Mesure la corrélation linéaire entre deux vecteurs.
- Chi-square (Chi carré) : Utilisée principalement pour les histogrammes.
- Intersection : Mesure l'intersection entre deux histogrammes.
- Bhattacharyya : Mesure la similarité entre deux distributions de probabilité.
- Brute force (pour ORB) : Recherche exhaustive des correspondances de points clés.
- FLANN (pour ORB) : Recherche rapide d'appariement de points d'intérêt basée sur des structures d'indexation.

5.3 Affichage des résultats

Top-k : L'utilisateur peut choisir d'afficher les Top-20, Top-50 ou Top-100 résultats les plus similaires à l'image requête.

Organisation : Les résultats sont présentés sous forme de grille, chaque image étant accompagnée de :

- Son rang,
- Son nom,
- Sa classe (animal/race),

- La distance ou le score de similarité,
- Les descripteurs utilisés,
- Une indication de pertinence (si la classe attendue est connue).

5.4 Évaluation des performances

Pour évaluer la qualité de la recherche, plusieurs métriques sont calculées automatiquement :

- Précision (Precision) : Proportion d'images pertinentes parmi les résultats retournés.
- Rappel (Recall) : Proportion d'images pertinentes retrouvées parmi toutes les images pertinentes de la base.
- F1-Score : Moyenne harmonique entre la précision et le rappel.
- Average Precision (AP) : Moyenne des précisions obtenues à chaque niveau de rappel.
- Mean Average Precision (mAP) : Moyenne des AP sur plusieurs requêtes (dans le SaaS, AP et mAP sont identiques pour une seule requête).
- R-Précision : Précision calculée sur le nombre total de documents pertinents (R) pour la requête.

Affichage : Les métriques sont affichées sous forme de tableaux et, si possible, une courbe Précision-Rappel est générée pour visualiser la performance du système.

A Exemples de résultats visuels (Interface et Résultats)

A.1 Page de connection

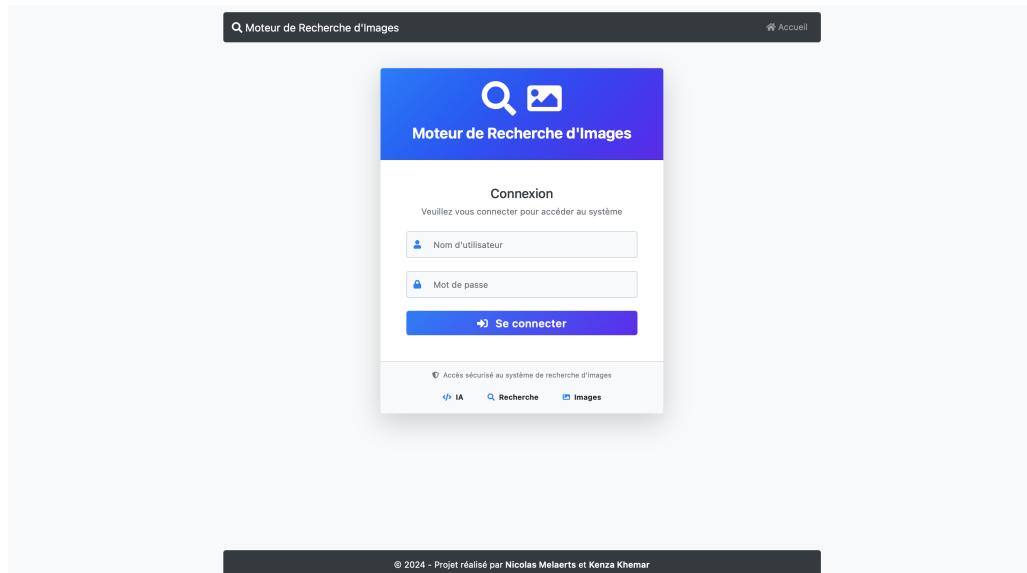


FIGURE 1 – Page d'accueil

A.2 Page d'accueil

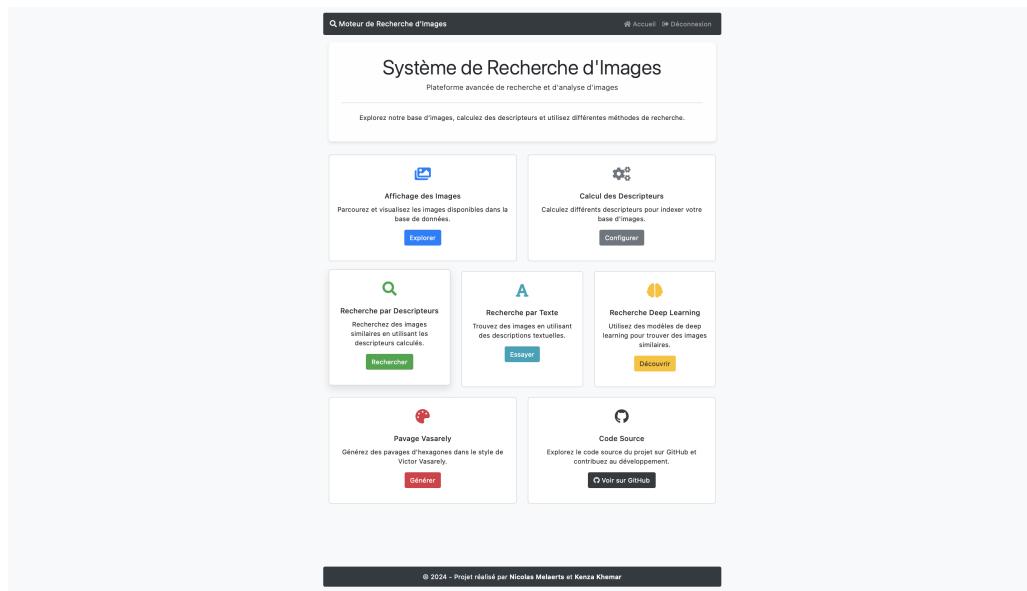


FIGURE 2 – Page d'accueil

A.3 Menu 1 : Affichage des images 1

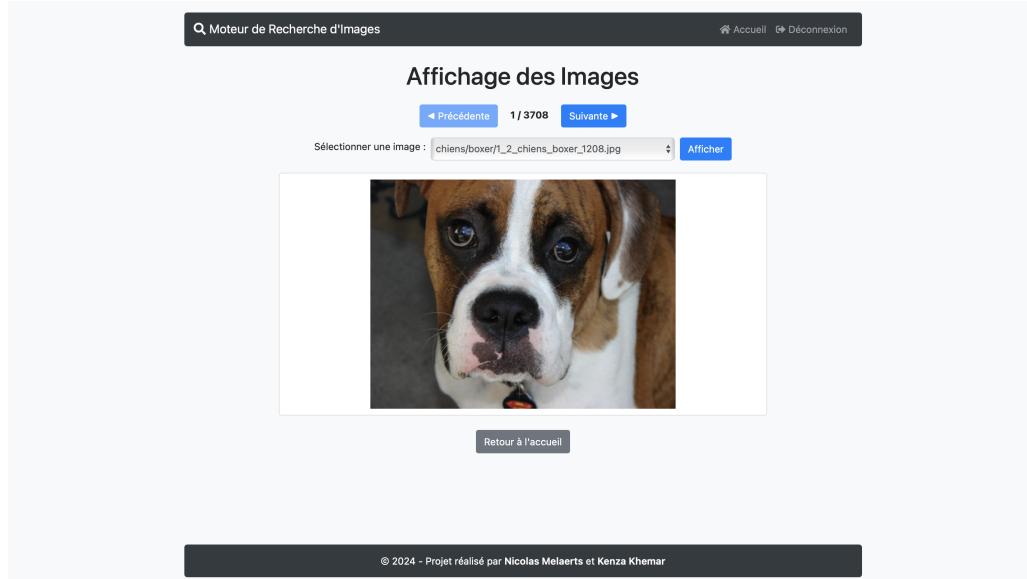


FIGURE 3 – Menu 1 : Affichage des images 1

A.4 Menu 1 : Calcul des descripteurs 1

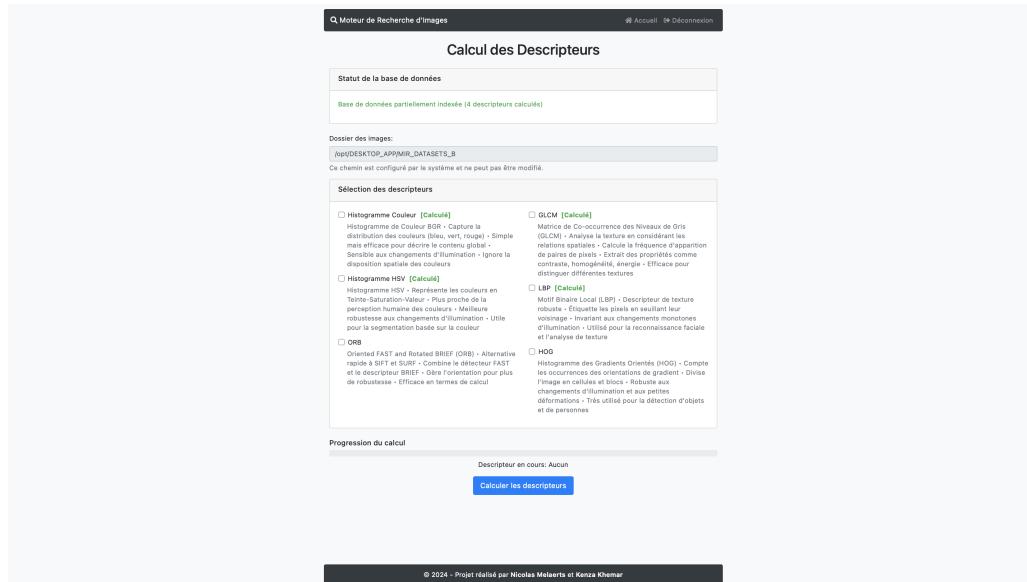


FIGURE 4 – Menu 1 : Calcul des descripteurs 1

A.5 Menu 3 : Recherche par descripteurs 1

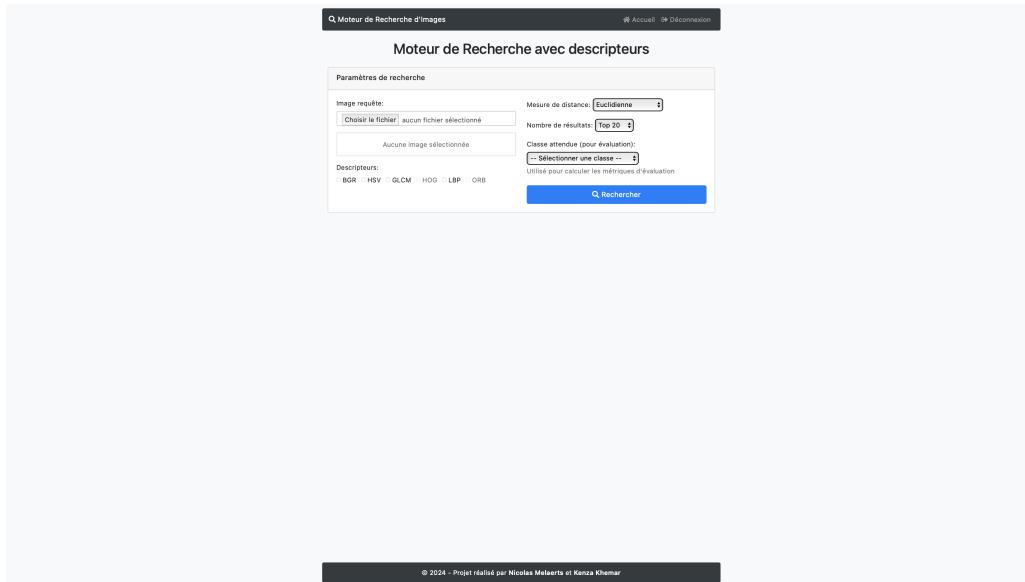


FIGURE 5 – Menu 3 : Recherche par descripteurs 1

A.6 Menu 3 : Recherche par descripteurs 2

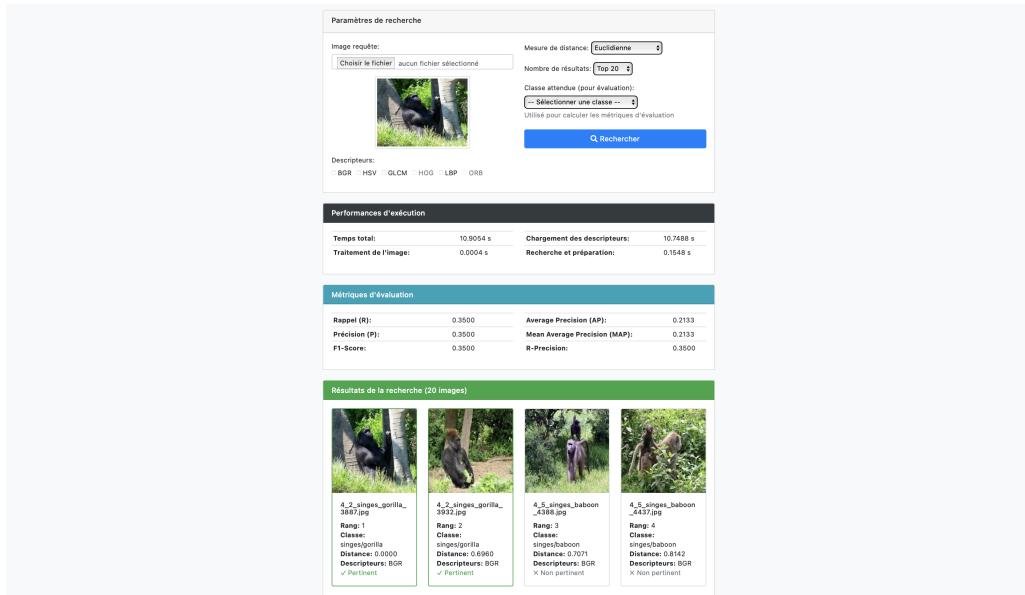


FIGURE 6 – Menu 3 : Recherche par descripteurs 2

A.7 Menu 4 : Recherche par texte 1

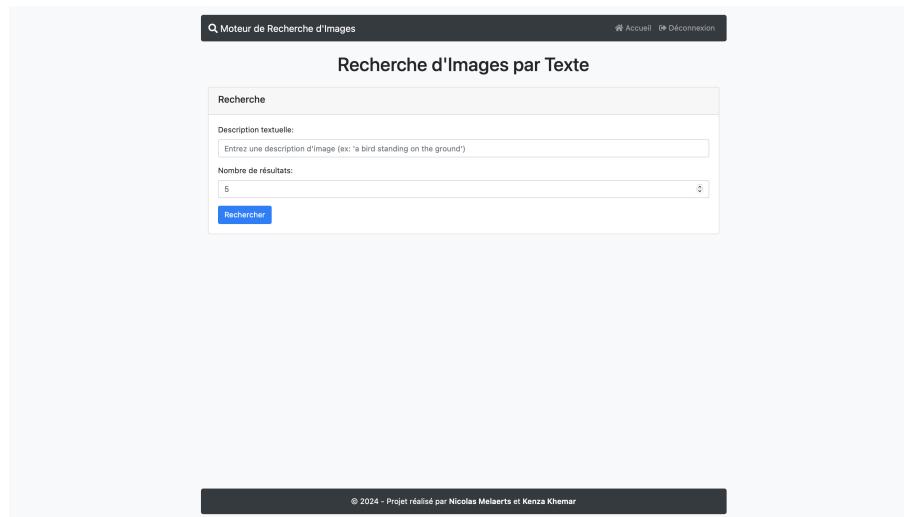


FIGURE 7 – Menu 4 : Recherche par texte 1

A.8 Menu 4 : Recherche par texte 2

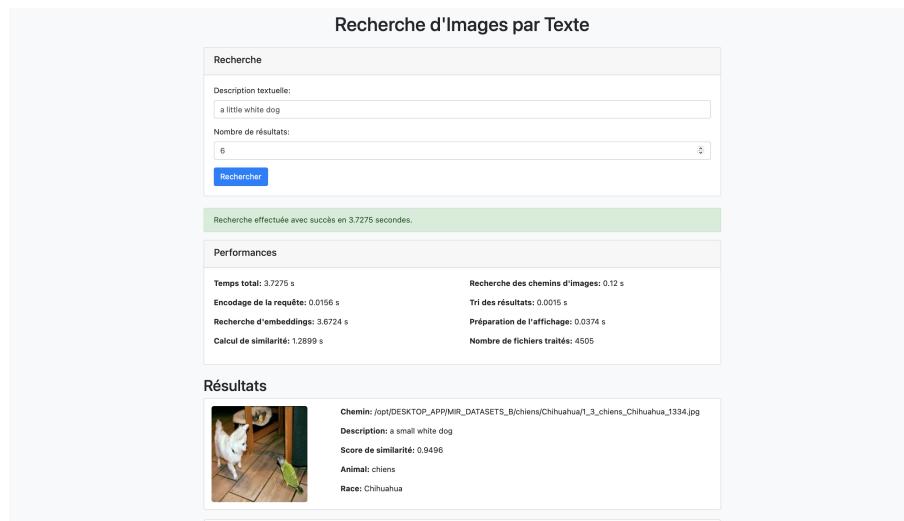


FIGURE 8 – Menu 4 : Recherche par texte 2

A.9 Menu 5 : Recherche Deep Learning 1

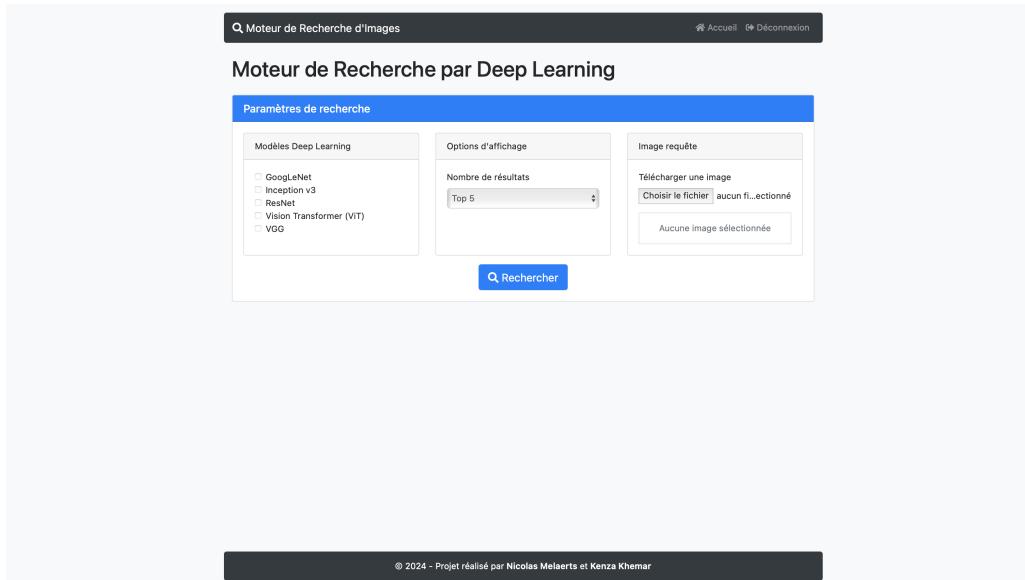


FIGURE 9 – Menu 5 : Recherche Deep Learning 1

A.10 Menu 5 : Recherche Deep Learning 2

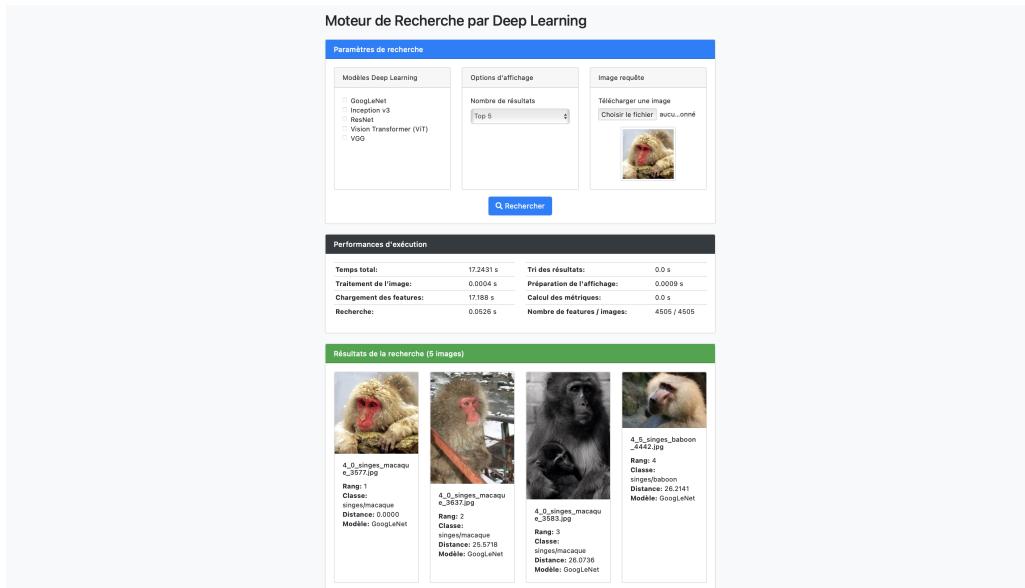


FIGURE 10 – Menu 5 : Recherche Deep Learning 2

A.11 Menu 6 : Génération Vasarely 1

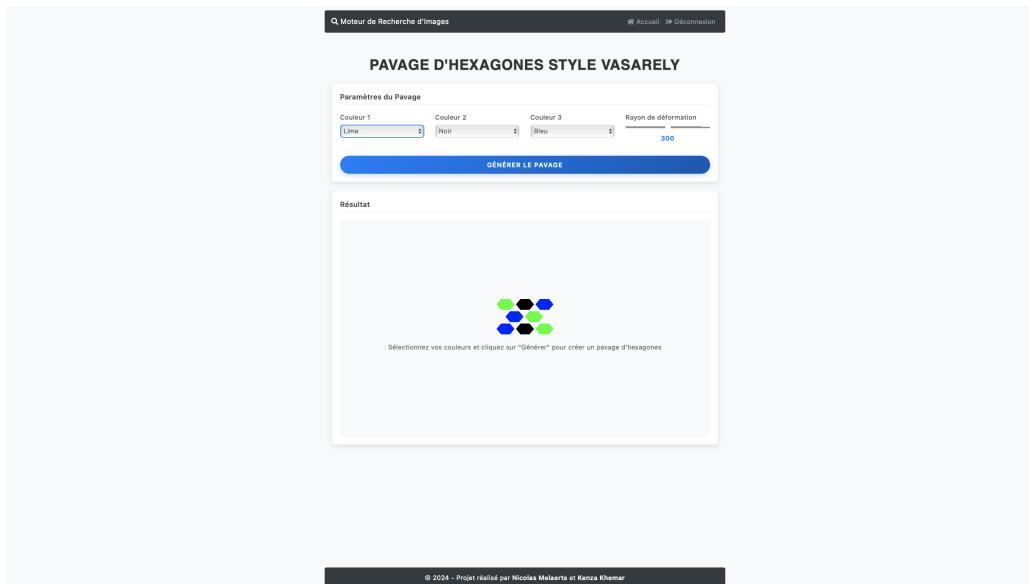


FIGURE 11 – Menu 6 : Génération Vasarely 1

A.12 Menu 6 : Génération Vasarely 2

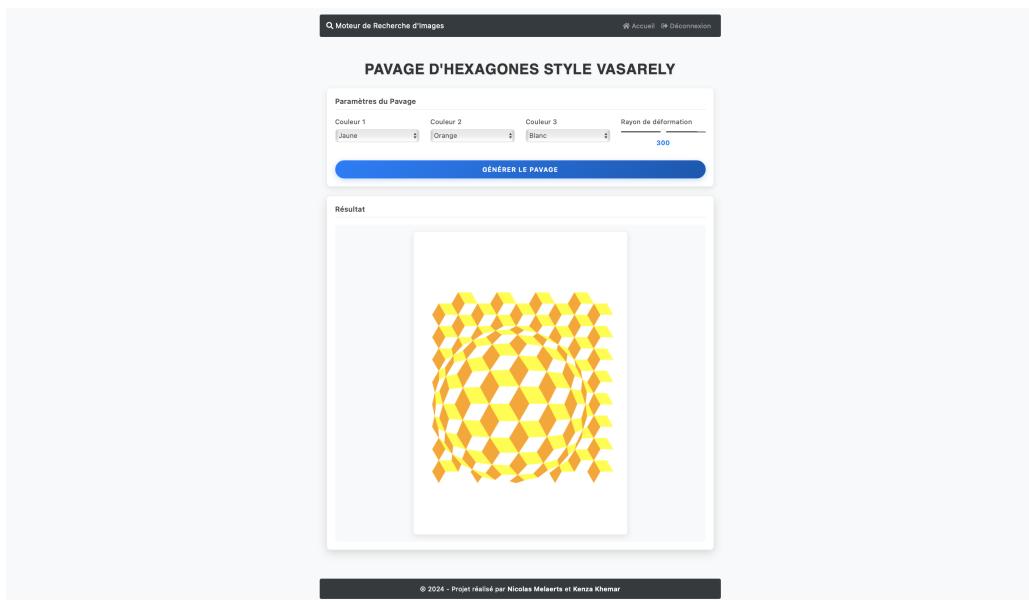


FIGURE 12 – Menu 6 : Génération Vasarely 2