Backend-модуль обратного поиска изображений

1. Общая задача

- Разработать backend-модуль обратного векторного поиска похожих людей по имеющемуся селфи.
- **Основной сценарий использования**: пользователь загружает фото с лицом, сервис находит похожие фотографии по базе.
- Модуль состоит из двух микросервисов:
    1. **Image-vectorizer** — векторизация изображений (поддерживает передачу `collection_id`).
    2. **Image-search** — поиск по базе (поддерживает поиск по всей базе или только по указанной коллекции).
- Хранение изображений — **Yandex Object Storage**.
- Хранение метаданных — **PostgreSQL** (доступ через **PostgREST**).
- Доступы к Object Storage и PostgREST предоставляются после начала работ.
- На момент старта загружено 30 изображений. В течение 2 дней после начала работ будут загружены ~10 000 изображений.

---

2. Нагрузки

- Поддержка **100 RPS** при поиске по двум коллекциям объёмом ~5 000 изображений каждая.
- Для тестирования будет предоставлена база **~10 000 изображений** из:
    - WIDERFACE
    - CelebA-Spoof

---

3. 📦 Стек технологий

Компонент	Технология
Язык	Python 3.10+
API-сервер	FastAPI

Работа с эмбеддингами	InsightFace
Хранение эмбеддингов	Milvus (pymilvus)
Деплой	Docker + Helm / Kubernetes

---

4. Технические параметры

4.1. Формат входных изображений

- Поддерживаемые форматы: **JPEG, PNG, WebP**.
- Лимиты размера вынесены в настройки. По умолчанию:
  - Поиск (`/search/`) — до **5 MB**.
  - Векторизация — до **20 MB**.
- Перед векторизацией изображения конвертируются в **sRGB**, **8 бит/канал**.

4.2. Параметры эмбеддингов

- Размер вектора (**dim**) — **512**, выносится в настройку.
- Нормализация — **L2-норма**, поиск с использованием **inner product (IP)**.
- Формат хранения — **float32**.

4.3. Параметры поиска

- Количество результатов (`top_k`):
  - По умолчанию — **10**.
  - Минимум — **1**, максимум — **50** (выносится в конфиг).
- Порог схожести (`score_threshold`) — **0.5** по умолчанию, настраиваемый.
- Поддержка пагинации — **да**.

4.4. Коллекции

- Задаётся при векторизации как `collection_id`.
- API для создания/удаления коллекций — **не требуется**.
- Если коллекция не указана — изображение сохраняется без коллекции.

---

5. Нефункциональные требования

- Упаковка сервисов в **Docker** и подготовка к деплою в **Kubernetes** (`deploy/k8s/` с манифестами или Helm-чартом).
- Обеспечение **идемпотентности данных**.
- Поддержка **zero-downtime**:
    - Стратегия деплоя — **RollingUpdate** (`maxUnavailable=0`, `maxSurge=1`).
    - Обязательные `readinessProbe` и `livenessProbe`.

---

6. 🛡️ Безопасность

- Доступ к API только при наличии корректного **X-Api-Key** (статический ключ из переменной окружения `API_KEY`).
- HTTPS обеспечивается на уровне **ingress**.
- Внутренний трафик между сервисами — HTTP.

---

7. ⚙ Эндпоинты

Метод	URL	Описание
GET	`/embed_status/`	Список изображений с индексами
POST	`/detect_faces/`	Возвращает массив `bbox` и `face_id`
PUT	`/add_face/`	Принимает `face_id` (или `photo_id + bbox`), генерирует и сохраняет эмбеддинг
DELETE	`/remove/{photo_id}`	Удаляет все лица с фото
DELETE	`/remove_all/`	Удаляет все лица всех фото
POST	`/search/`	Поиск по селфи, вывод уникальных изображений (при нескольких лицах — самое похожее)
GET	`/ping`	Проверка доступности сервиса

Content-type: `multipart/form-data`

---

8. Формат ответа `/search/`

json

```
[
  {
    "photo_id": "123",
    "face_id": "123",
    "collection_id": "event_456",
    "score": 0.92,
    "bbox": [120, 80, 200, 160],
    "storage_key": "events/2025-08-28/photo_001.jpg"
  }
]
```

---

9. Обработка ошибок

- `"too_many_faces"` — на фото более одного лица.
- `"no_faces_detected"` — лицо не найдено.
- `"invalid_image_format"` — неподдерживаемый или повреждённый файл.

---

10. Тестирование

- **Нагрузочное тестирование**:
  - Набор 5000 изображений в Object Storage и Milvus.
  - `/search/` с `top_k=10` и `top_k=50` при **RPS 100** в течение 5 минут.
  - Метрики: среднее время ответа, 95-й и 99-й перцентиль.
  - Инструменты: **Locust** или **k6**.
- Проверка работы на **пустых коллекциях** — возврат пустого массива без ошибок.

---

11. Развёртывание и масштабирование

- Инфраструктура: **Kubernetes on-prem**.
- Поддержка **HPA (Horizontal Pod Autoscaler)**:
  - Порог CPU/памяти настраиваемый.
  - Min/max подов — настраиваемые.
  - Обеспечивать стабильную работу при пиках до **100 RPS**.
- Очереди для поиска **не используются**, запрос `/search/` выполняется синхронно.

---

12. Приёмка

1. Деплой:

```
kubectl apply -f deploy/k8s/
# или
helm upgrade --install image-search deploy/k8s/
```

2. Проверка, что Pods запущены без ошибок.
3. Выполнение тестов по всем эндпоинтам.
4. Запуск нагрузочного теста **Locust** или **k6**.