API v0 – przewodnik użycia

Ten dokument opisuje, jak korzystać z API v0 do znajdowania konkretnej osoby (twarzy) w zbiorze zdjęć dostępnych pod adresami URL. Silnik wykrywa twarze i liczy wektory cech (embeddingi), a wyniki są cache'owane na dysku na podstawie hasha URL.

Domyślny adres serwisu: `http://localhost:5003`

Szybki start (zalecany przepływ)

- 1. Wstępnie zcache'uj obrazy: wyślij wszystkie adresy URL, z którymi będziesz pracować (zarówno `target`, jak i `scope`) do `POST /api/v0/embed`.
- 2. (Opcjonalnie) Podejrzyj twarze w obrazie celu: użyj `POST /api/v0/inspect`, aby poznać indeksy wykrytych twarzy i wybrać `target_face`.
- 3. Wyszukaj osobę w zbiorze: wywołaj `POST /api/v0/findIn`, przekazując `target` i `scope`, z opcjonalnymi parametrami (`threshold`, `target_face`, `include_details`, `max_results`).

Endpointy API v0

POST `/api/v0/embed` – wstępne cache'owanie obrazów

Wczytuje obrazy spod podanych adresów URL, wykrywa twarze i zapisuje embeddingi w cache. Dzięki temu późniejsze zapytania nie muszą ponownie liczyć wektorów.

- Body (JSON):

```
{
  "urls": ["https://.../obraz1.jpg", "https://.../obraz2.jpg"]
}
```

- Przykład (curl):

```
curl -X POST "http://localhost:5003/api/v0/embed" \
  -H "Content-Type: application/json" \
  -d '{
    "urls": ["https://example.com/a.jpg", "https://example.com/b.jpg"]
  }'
```

- Przykładowa odpowiedź (200):

```
{
  "success": true,
  "total_urls": 2,
  "results": [
```

```
    {
      "url": "https://example.com/a.jpg",
      "success": true,
      "cached": true,
      "num_faces": 1,
      "cache_file": "<ścieżka_do_pliku_cache>"
    },
    {
      "url": "https://example.com/b.jpg",
      "success": true,
      "cached": true,
      "num_faces": 3,
      "cache_file": "..."
    }
  ]
}
```

POST `/api/v0/inspect` – podgląd wykrytych twarzy

Zwraca listę wykrytych twarzy w obrazie (indeks, bbox, score), aby ułatwić wybór konkretnej twarzy do dopasowania.

- Body (JSON):

```
{ "url": "https://example.com/target.jpg" }
```

- Przykład (curl):

```
curl -X POST "http://localhost:5003/api/v0/inspect" \
  -H "Content-Type: application/json" \
  -d '{
    "url": "https://example.com/target.jpg"
  }'
```

- Przykładowa odpowiedź (200):

```
{
  "success": true,
  "url": "https://example.com/target.jpg",
  "faces_count": 2,
  "faces": [
    { "index": 0, "bbox": [x1, y1, x2, y2], "score": 0.99 },
    { "index": 1, "bbox": [x1, y1, x2, y2], "score": 0.95 }
  ]
}
```

POST `/api/v0/findIn` – wyszukiwanie osoby w zbiorze obrazów

Porównuje twarz(e) z obrazu `target` z twarzami na obrazach w `scope` i zwraca uporządkowane dopasowania.

- Body (JSON):

```
{
  "target": "https://example.com/target.jpg",
  "scope": [
    "https://example.com/photo1.jpg",
    "https://example.com/photo2.jpg"
  ],
  "threshold": 0.6,
  "target_face": "all",
  "include_details": false,
  "max_results": 50
}
```

- Przykład (curl):

```
curl -X POST "http://localhost:5003/api/v0/findIn" \
  -H "Content-Type: application/json" \
  -d '{
    "target": "https://example.com/target.jpg",
    "scope": ["https://example.com/p1.jpg", "https://example.com/p2.jpg"],
    "threshold": 0.65,
    "target_face": "best",
    "include_details": true,
    "max_results": 100
  }'
```

- Przykładowa odpowiedź (200):

```
{
  "success": true,
  "target_url": "https://example.com/target.jpg",
  "target_faces_count": 2,
  "threshold": 0.65,
  "total_scope_images": 2,
  "total_matches": 2,
  "urls": ["https://example.com/p1.jpg", "https://example.com/p2.jpg"],
  "matches": [
    {
      "url": "https://example.com/p1.jpg",
      "similarity": 0.84,
      "target_faces_found": 1,
      "target_face_indices": [0],
      "face_matches": [
```

```
        {
          "target_face": 0,
          "scope_face": 1,
          "similarity": 0.84,
          "target_bbox": [x1, y1, x2, y2],
          "target_score": 0.99,
          "scope_bbox": [x1, y1, x2, y2],
          "scope_score": 0.97
        }
      ],
      "scope_faces_count": 3
    },
    {
      "url": "https://example.com/p2.jpg",
      "similarity": 0.66,
      "target_faces_found": 1,
      "target_face_indices": [1]
    }
  ],
  "selected_target_indices": [0, 1],
  "target_summary": [
    { "index": 0, "bbox": [x1, y1, x2, y2], "score": 0.99 },
    { "index": 1, "bbox": [x1, y1, x2, y2], "score": 0.95 }
  ]
}
```

Pola `face_matches`, `scope_faces_count`, `selected_target_indices` i `target_summary` pojawiają się, gdy `include_details` = `true`.

Dodatkowe endpointy pomocnicze

GET `/api/health`

Szybki test zdrowia usługi.

```
curl "http://localhost:5003/api/health"
```

GET `/api/cache/stats`

Statystyki cache (np. liczba wpisów).

POST `/api/cache/clear`

Czyści cały cache embeddingów.

POST `/api/cache/cleanup`

Usuwa nieprawidłowe wpisy cache.

Parametry i wskazówki

- **threshold (0..1, domyślnie 0.6)**: minimalne podobieństwo kosinusowe, aby uznać parę twarzy za dopasowaną. 0.5–0.6 to dobry start.
- **target_face**:
  - `"all"` – użyj wszystkich twarzy,
  - `"largest"` – największa twarz (po polu bbox),
  - `"best"` – twarz z najwyższym `det_score`,
  - liczba (np. `0`) lub lista liczb (np. `[0,2]`) – konkretne indeksy.
- **include_details (bool)**: jeśli `true`, odpowiedzi zawierają bbox-y, score-y i parowania twarz–twarz.
- **max_results (int)**: ogranicza liczbę zwracanych wyników.
- **Cache**: Jeśli chcesz przyśpieszyć wyszukiwanie to wrzuc zdjęcia do `POST /api/v0/embed`

Obsługa błędów

Format błędu:

```
{
  "error": "Opis błędu",
  "success": false
}
```

Kody HTTP:

- `400` – niepoprawne żądanie (brak pól, zły typ danych, threshold poza zakresem),
- `500` – błąd wewnętrzny (problem z pobraniem obrazu, przetwarzaniem, I/O).

Uruchomienie i uwagi praktyczne

- Serwis nasłuchuje na porcie `5003`.
- Włączone jest CORS (można wywoływać API z przeglądarki podczas developmentu).
- Dla dużych obrazów czasy przetwarzania mogą wzrosnąć; jakość zdjęć wpływa na stabilność wyników.

Notatka

Jeśli pracujesz na plikach lokalnych (katalog `data/`), istnieje starszy endpoint `POST /api/findIn` działający na lokalnym katalogu. Ten dokument opisuje ścieżki `api/v0/*`, oparte na adresach URL.

TL;DR (rekomendacja)

Zaraz po wgraniu zdjęć na serwer zrób cache przez `POST /api/v0/embed`. Dzięki temu w `POST /api/v0/findIn` embeddingi będą już policzone i zapisane na dysku (hash URL), co znacząco przyspiesza wyszukiwanie.